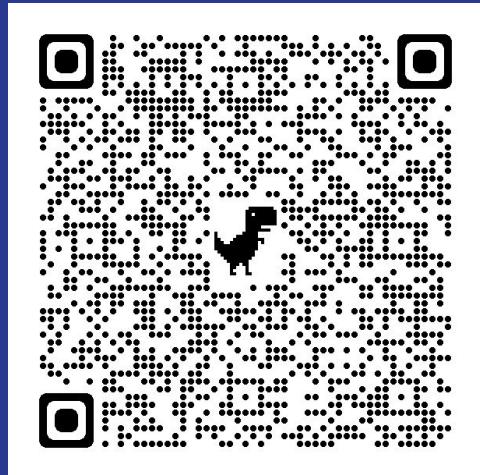


# Platform Engineering

Mathieu Benoit  
Customer Engineer @ Humanitec

[medium.com/@mabenoit](https://medium.com/@mabenoit)

Agile Québec - September 7th, 2023



*Updated version on 2023-09-14*



# whoami

Cloud Native Security

Agile / Scrum / XP

ASP.NET

Docker / Kubernetes

Platform Engineering

Java / UNIX

.NET Windows

Microsoft Azure

DevOps

Terraform

Google Cloud

AWS

2004

2008

2010

2014

2016

2020

2023

Software Engineer

Platform Enabler

Continuous Learner

# Agenda

Lessons learned and success criteria for your own Platform Engineering practice

What's DevOps?

What's Platform Engineering?

What makes a Platform successful?

Which tools and technologies to use?

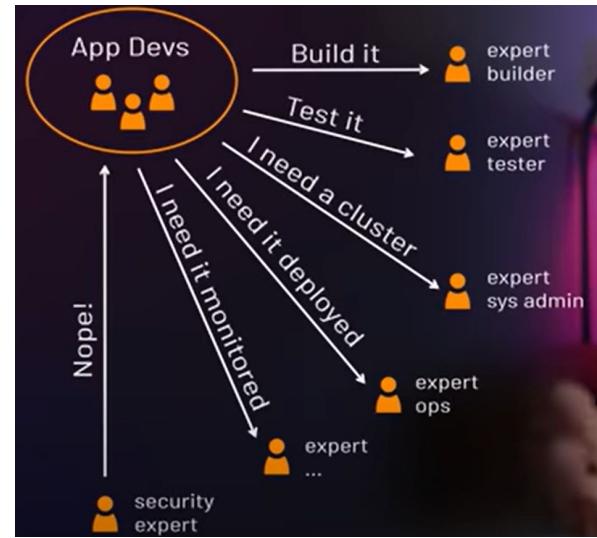
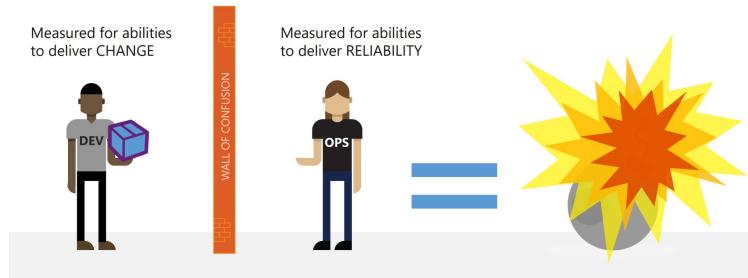
Quick demo!

Where to start?

Q&A

---

# Where DevOps is coming from?



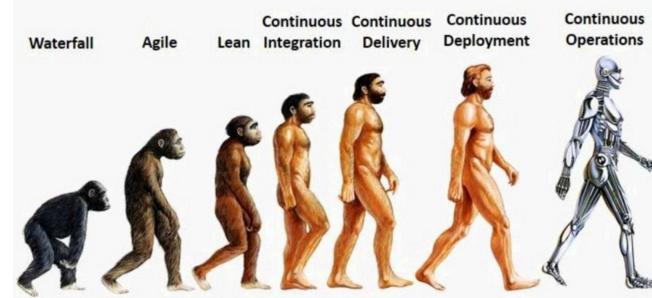
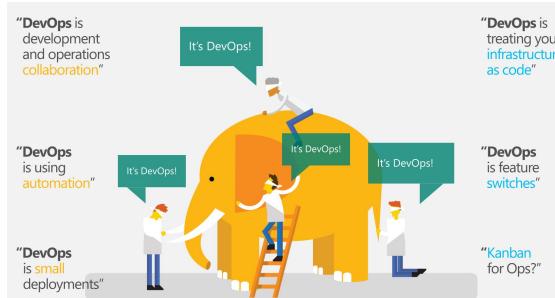
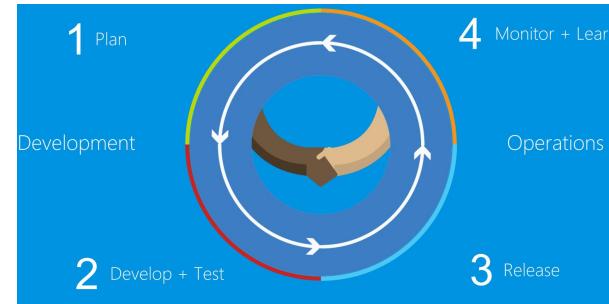
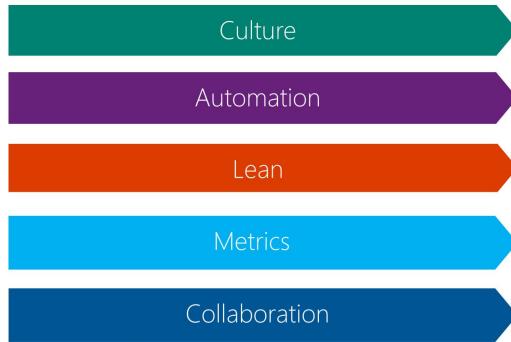
[DevOps Is Dead! Long Live Platform Engineering! Did We Get Confused? - Viktor Farcic](#)



# What is DevOps?

DevOps Mindset (2009): [The rise of the DevOps mindset - Stack Overflow](#)

5 pillars



*“The DevOps movement is built around a group of people who believe that the application of a **combination** of appropriate **technology** and **attitude** can revolutionise the world of **software development and delivery.**” - Patrick Debois (2010)*

## How developers spend their time



[How Much Time Do Developers Spend Actually Writing Code? - The New Stack](#)

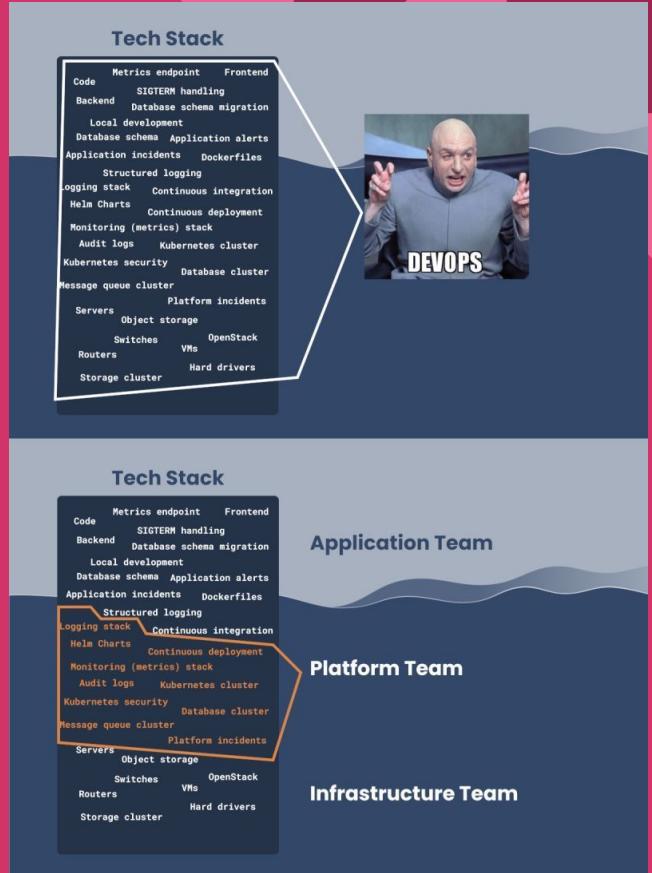
## Looking back at my dev career Cognitive Load

Year (Approx)	App Architecture	Infra / Fabric	My (Developer) Responsibility	Developer Control Planes
2000	Monolith	In-house tin	Code	IDE, CVS, deploy portal
2005	Monolith / SOA	In-house / colo / cloud	Code, ship, [limited run]	IDE, Mercurial, Jenkins, [PXE, bash, Puppet]
2010	Monolith	Heroku / CF	Code, run	IDE, Git, Heroku CLI, Heroku UI, New Relic UI
2015	Microservices	Cloud	Code, ship, run	IDE, Git, Docker Hub, Jenkins+plugins, AWS Console, bash, Terraform, Chef...
2020	Microservices++	K8s	Full lifecycle (code, ship, run)++	IDE, Git, K8s

[From Kubernetes to PaaS to ... err, what's next? • Daniel Bryant • PlatformCon 2022](#)

*“You build it, you run it.”*  
- Amazon’s CTO  
Werner Vogels (2006)

# DevOps is dead, long live Platform Engineering!



# Platform and Program Teams

***"After a huge amount of data collecting, thinking and debating among many folks across the company, we are ready to launch Programs & Platforms! (Attached to this email) you'll find out whether you're on a Program or Platform team, and where your seat will be with your new team."*** - [Uber \(2010\)](#)

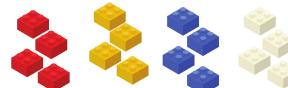
## Program

*long-lived, cross-functional (BE, FE, AI, etc.), external customers, focused on a business mission*



## Platform

*focused on a technical mission, specialized and rarely cross-functional, customers are engineering teams and are internal, consumed by multiple verticals (programs and platforms)*

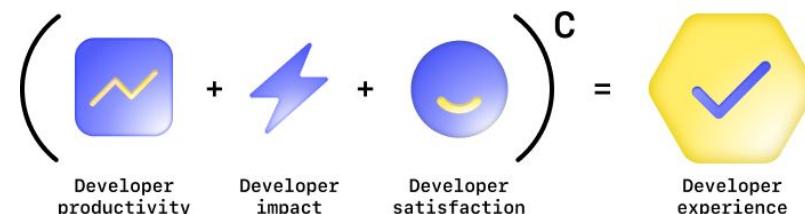


# Developer eXperience - reducing the cognitive load

*“The cognitive load involved in a task is the **cognitive effort** (or amount of information processing) required by a person to perform this task.” - [Reif \(2010\)](#)*

*“Developer eXperience is about creating an **environment** in which a developer can do their **best work**.” - [James Governor from Redmonk \(2022\)](#)*

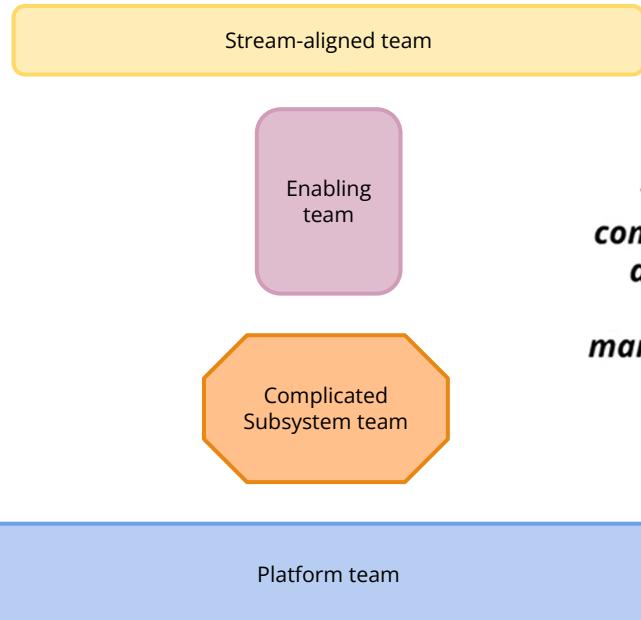
The collaboration effect on developer experience



[Developer experience: What is it and why should you care? - The GitHub Blog](#)

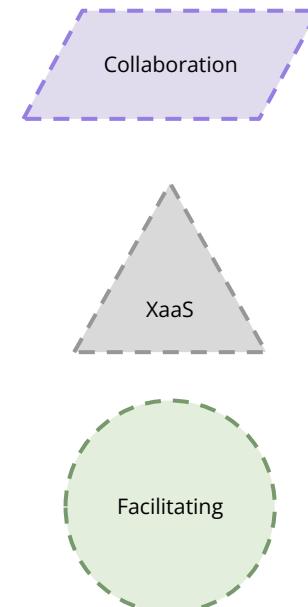
# Organizing Teams for fast flow

## 4 Team Types



*"Highly evolved firms use a **combination of stream-aligned and platform teams** as the most effective way to **manage cognitive load at scale**"*

## 3 Interaction Modes





# What makes a Platform successful?

# What's a Platform?

*“A digital platform is a **foundation of self-service APIs, tools, services, knowledge and support** which are arranged as a compelling internal product. Autonomous delivery teams can make use of the platform to **deliver product features at a higher pace, with reduced coordination.**” - [Evan Bottcher](#)*



# Platform as Product



We've spent months building this platform, devs hate it, help me understand why

on a deployment by deployment basis. Also takes care of Secrets etc.. Honestly, we are pretty proud of the result and does the job. Our team was excited, EMs too. We spent a lot of time explaining the platform to devs and why it's good for them i.e. they can now self-serve everything end to end without being dependent on us.

Then devs just say "no". There's no clear reasoning here, they don't say what is wrong or why they would refuse, they cannot point at any case that we do not cover and there is an easy and clear way of

[https://www.reddit.com/r/sre/comments/stuekd/weve\\_spent\\_months\\_building\\_this\\_platform\\_devs/](https://www.reddit.com/r/sre/comments/stuekd/weve_spent_months_building_this_platform_devs/)

Value proposition, Org/Company alignment

Clear roadmap/backlog

Ongoing attention, evolve and adapt based on developer feedback and the changing business landscape

Product Manager/Owner

User research (quantitative and qualitative)

Alpha/Beta/GA features

Support and SLOs

Platform Advocate/Marketing - have a brand

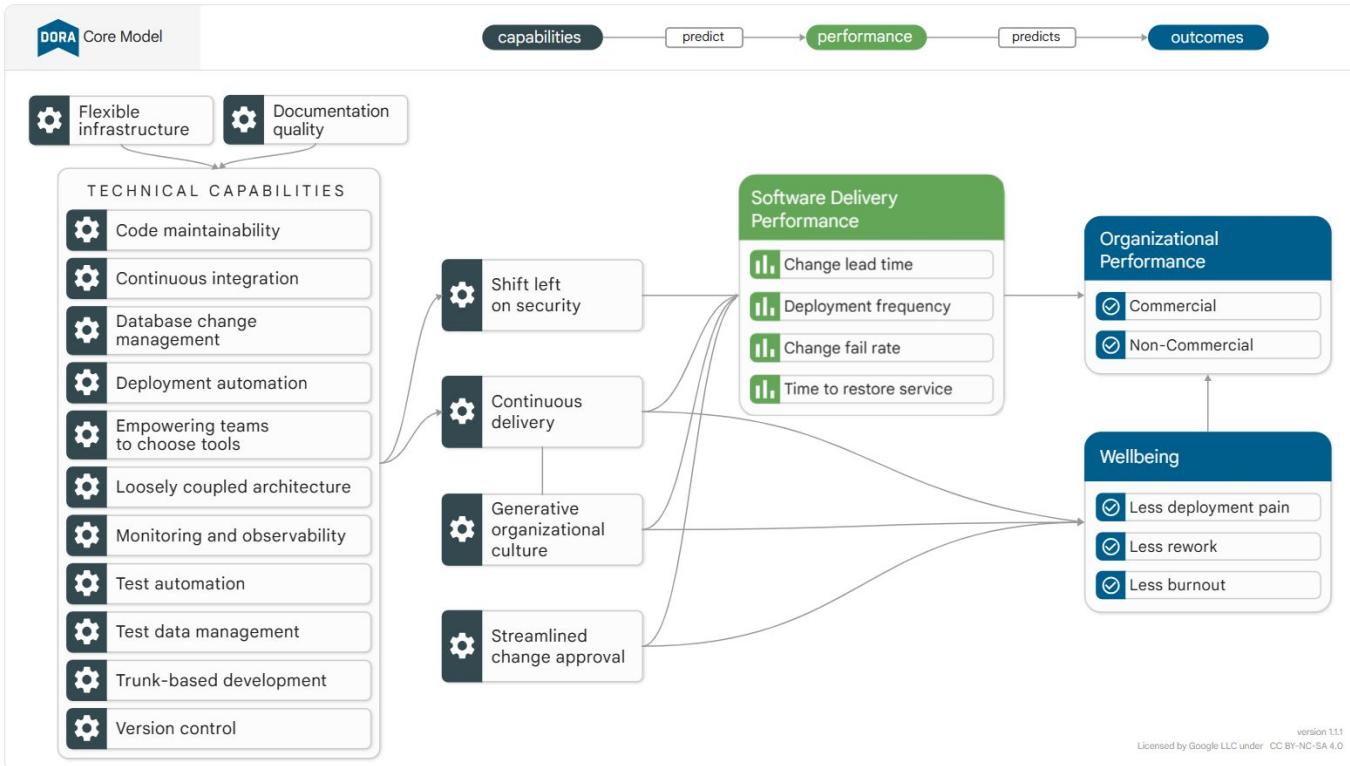
Measure adoption and usage

# Pave Golden Paths to drive standardization, and less frictions

- Describe opinionated and **well-supported paths** to “build something”
- Reduce set of **tools and technology preferred** to deliver values to the business
- Focus on an **engineer’s intentions**, not making engineers worry over implementation details

*“The Golden Path — as we define it today — is the ‘opinionated and supported’ path to ‘build something’ (for example, build a backend service, put up a website, create a data pipeline). The Golden Path tutorial is a **step-by-step tutorial** that walks you through this opinionated and supported path” - [Spotify \(2014\)](#)*

# Speed, Productivity and Stability



# SPACE

5 dimensions published in 2021:



+ Think also about: OKRs, FinOps, SLOs

# The Platform Maturity Model

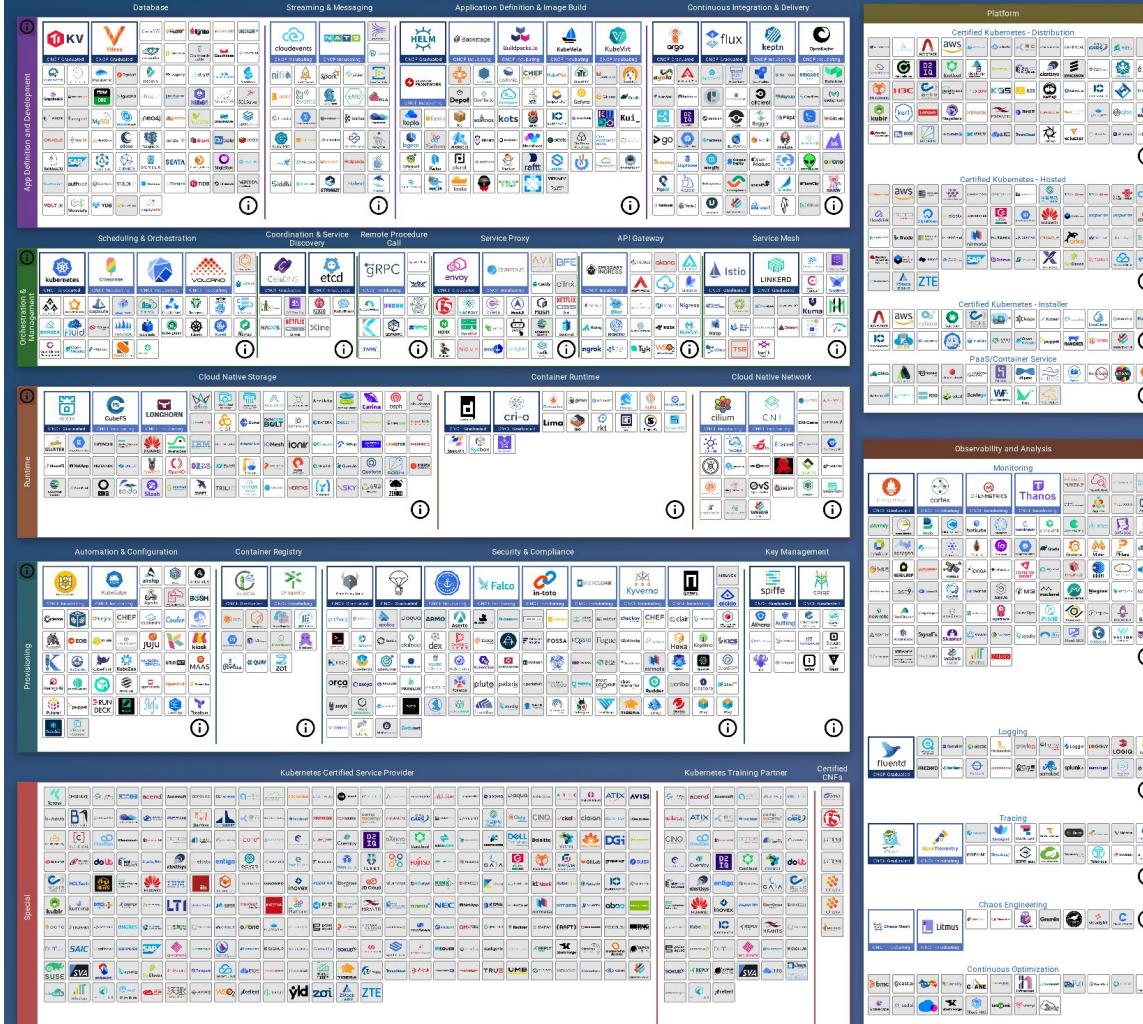
[Syntasso donates first version of Platform Maturity Model to CNCF Working Group](#)

[Platform engineering maturity model: what Humanitec learned from its survey of ~300 orgs](#)

	1	2	3	4	
Funding	How does the company value (and therefore fund) platform efforts?	One-off	Annual platform budget	Platform team budget	Profit and loss
Adoption	What compels users to start, and be successful, using your platform?	Mandatory	Build it and they will come	Internal champions	Platform advocacy
UX	How do users interact with and consume offerings from your platform?	Manual request queue	Off-the-shelf offerings	Curated entry point	Paved paths
Backlog	How are requests and requirements identified and prioritized for your platform?	Reactive	Scheduled	Evolutionary	Platform as a Product
Organizational structure	How does product engineering manage non-differentiating (and often internally common) tooling and infrastructure?	Dev and Ops	Full stack developers	Developer enablement	Platform team(s)
Cross-functional representation	How does each business requirement (e.g., compliance or performance) get enabled by platform offerings?	Off platform	Tools provided	Automated by platform team(s)	Specialists driven



# Which tools and technologies to use?



## . CONTAINERIZATION

- Commonly done with Docker containers  
Any size application and dependencies (even PDP-11 code running on an emulator) can be containerized  
Over time, you should aspire towards splitting suitable applications and writing future functionality as microservices



## 2. CI/CD

- Set up Continuous Integration/Continuous Delivery (CI/CD) so that changes to your source code automatically result in a new container being built, tested, and deployed to staging and eventually, perhaps, to production
  - Set up automated rollbacks, roll backs and testing
  - Argo is a set of Kubernetes-native tools for deploying and running jobs, applications, workflows, and events using GitOps paradigms such as continuous and progressive delivery and MLOps

### **E. ORCHESTRATION & APPLICATION DEFINITION**

- Kubernetes is the market-leading orchestration solution. You should select a Certified Kubernetes Distribution, Hosted Platform, or Installer: [cncf.io/ck](https://cncf.io/ck). Helm Charts help you define, install, and upgrade even the most complex Kubernetes application.



## 5. SERVICE PROXY, DISCOVERY, & MESH

- CoreDNS is a fast and flexible tool that is useful for service discovery. Envoy and Linkerd each enable service mesh architectures. They offer health checking, routing, and load balancing.



## **D. DISTRIBUTED DATABASE & STORAGE**

Then you need more resiliency and scalability than you can get from a single database. Vitess is a good option for running MySQL at scale through sharding. CockroachDB is a storage orchestrator that integrates a diverse set of storage solutions into Kubernetes. It serves as the ‘brain’ of Kubernetes, and it provides a reliable way to store data across a cluster of machines. KV is a high-performance distributed transactional key-value store written in Rust.



6. NETWORKING, POLICY,  
& SECURITY

To enable more flexible networking, use a CNI-compliant network project like Calico, Flannel, or Weave Net. Open Policy Agent (OPA) is a general-purpose policy engine with uses ranging from authorization and admission control to data filtering. Falco is an anomaly detection engine for cloud native.



## STREAMING & MESSAGING

If you need higher performance than JSON-REST, consider gRPC or NATS. gRPC is a universal RPC framework. NATS is a multi-modal messaging system that includes request/reply, pub/sub and load balanced queues. CloudEvents is a specification prescribing event data in common ways.



CONTAINER REGISTRY & RUNTIME

Container is a registry that stores, signs, and scans content. You can use alternative container runtimes. The most common, both of which are OCI-compliant, are containerd and CRI-O.

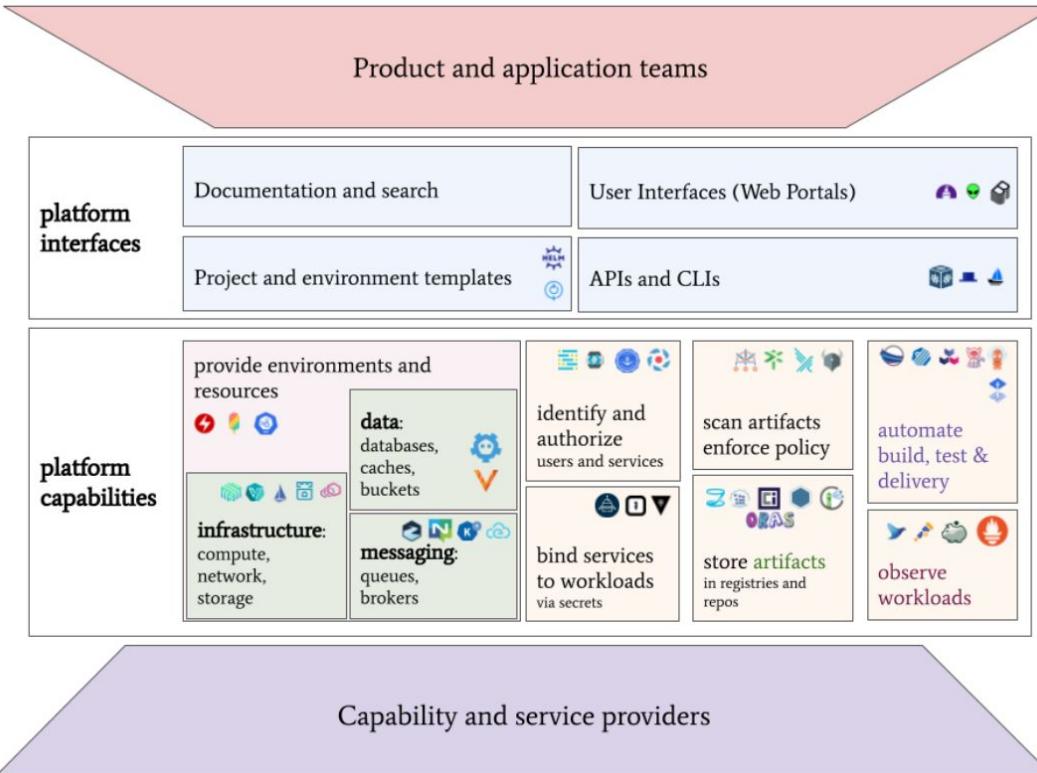


## 9. SOFTWARE DISTRIBUTION

you need to do secure software distribution, evaluate Notary, an implementation of The Update Framework.



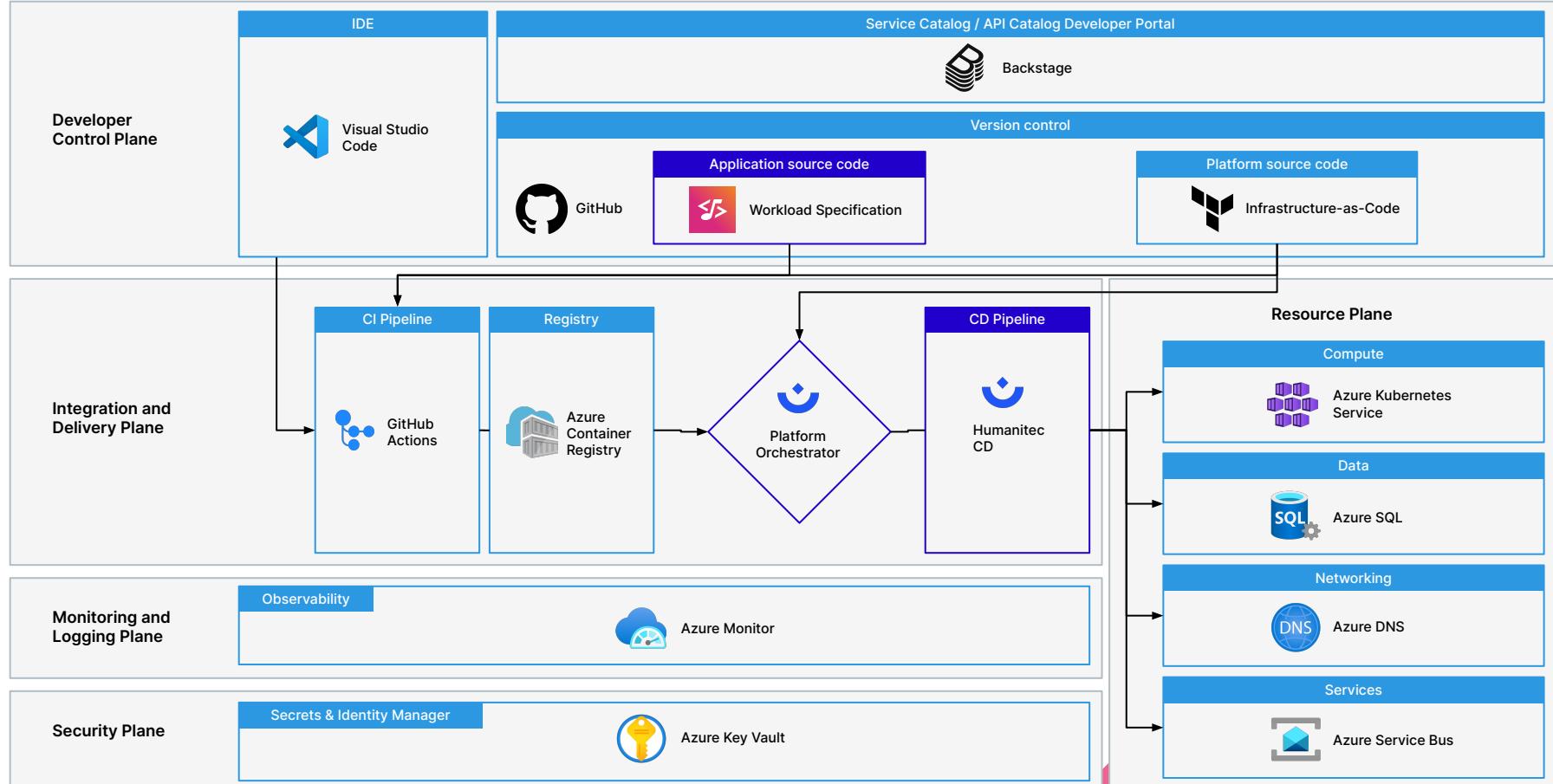
# Capabilities of a Platform



# Demo time!



# Example of an Internal Developer Platform on Azure Cloud with Humanitec



# Example of a Golden path: As Developer, Deploy my feature branch to an ephemeral environment

Dev  
Request

```
score.yaml |  
  
apiVersion: score.dev/v1b1  
  
metadata:  
  name: python-service  
  
containers:  
  python-service:  
    image: python  
    variables:  
      CONNECTION_STRING: postgresql://${resources.db.user}:...@${resources.db.host}/...  
  
resources:  
  do:  
    type: postgres  
  storage:  
    type: s3  
  dns:  
    type: dns
```

context:  
**env=ephemeral**



Platform  
response

- ✓ **Read** workload specification
- ✓ **Match** resource definitions
- ✓ **Create** app configs, configure resources
- ✓ **Deploy**

- AKS cluster configured
- Kubernetes namespace created
- Create PostgreSQL database
- PostgreSQL credentials injected
- DNS configured

Workload deployed with dependencies injected



That's a wrap!

# Summary

Like [how Chad McElligott summarizes what Platform Engineering is](#):

- What: **Velocity**
- Why: **Stability**
- How: **Product Mindset**

In addition to that:

- Invest in the **Developer eXperience**.
- Avoid frictions and accelerate adoption with well-supported **Golden Paths**.
- Invest in your **Enabling Team**, think about Developer Advocates/Evangelists as internal champions.

# Where to start?

Set a mission and clarify goals, have success criterias

*Think OKRs!*

Truly know your Customers/Developers

*Think about Customer User Journeys (CUJs) or Golden Paths!*

Provide boundaries and abstractions to reduce the cognitive loads on teams

Adopt a Platform as Product mindset

*Think OSS/Innersource projects!*

UX first: Developer Experience & Product Experience

Find the first right level of abstraction, start small with high impact

Measure quality, productivity, getting feedback

*Think surveys!*

Use your own Platform to build your own Platform's services

*Think dogfooding!*

# That's not it, here are more tips!

- That may not be your concern if you are 1 team/1 person/1 product! ;)
- Tie the **measurable goals** to business outcomes
- Don't assume that if you build it they will come
- Start small, don't be afraid to **fail fast** (Tight feedback loops)
- Avoid too many tools or workflows
- Provide **self-service**
- **Request for Comments (RFC) process**
- Regarding to the **golden paths**, think about both:
  - onboarding/new
  - tenured employees/projects
- Communicate and **celebrate**
- Measure **adoption** and **satisfaction**
- Invest in your team: **PM, PO, UX, SRE**
- Have a **brand**, and stickers/t-shirts
- Don't forget that **Kubernetes** is a platform, but you need to build your platform on top of it
- The success of an internal platform is defined by how many teams **adopt** it
- Just because you build it, does not mean that they will come. **You must go to them.**
- An internal developer platform should not be a catchall

Platform Engineering Mission

BUILD THE FOUNDATIONS THAT  
ENABLE TEAMS TO SHIP CONFIDENTLY;  
QUICKLY & EFFICIENTLY, WITHOUT  
COMPROMISING STABILITY

Lambros Charassis  
Senior Technical Product Manager  
Wise

Productivity	Stability	Efficiency	Risk
--------------	-----------	------------	------

**Core Principles**  
Our approach to building at scale

**System analysis**  
With a system-level view and a user-centric view, we work hard to identify gaps and inefficiencies in our engineering process so that we can build solutions to improve engineering excellence and velocity.

**Instrumentation**  
We believe that you can't improve what you can't measure. Google is a data-driven company and we are a data-driven discipline. We obsess over metrics and work hard to move them in the right direction.

**Tools and infrastructure**  
We build critical tools and infrastructure to help Google engineers work more effectively and efficiently. This enables Google to ship excellent products, faster.

**Focus on the user**  
We embed in product engineering teams where we champion polished products for Google's users and fast, scalable engineering for our users. Google's engineers.

[landing.google.com/engprod](http://landing.google.com/engprod)

# More resources

[platformengineering.org](http://platformengineering.org)

[platformcon.com](http://platformcon.com)

[Platform Engineering: The New Stack's ebook](#)

[Accelerate \(2018\)](#)

[How to write a talk about Team Topologies](#)

[How to measure and improve developer productivity](#)

[Platform engineering is just DevOps with a product mindset](#)

[How Spotify Achieved a Voluntary 99% Internal Platform Adoption Rate](#)

[Platform engineering maturity model: what we learned from our survey of ~300 orgs](#)

[DevOps and Kubernetes: we've been doing it wrong](#)

[DevOps is Bullshit](#)

[internaldeveloperplatform.org](http://internaldeveloperplatform.org)

[Beyond engineering: The future of platforms](#)

[Quantifying platform engineering impact at Wise](#)

[A conversation on platform engineering: What, how, and what's next](#)

[On growth challenges, generic helm charts and golden paths](#)

[How Platform Engineering Works](#)

[Team Topologies Distilled](#)

[Fidelity's Software Delivery Platform](#)

[Platform as a Product Workshop](#)

[From Kubernetes to PaaS to ... err, what's next?](#)

# Q&A

Merci !  
Thank you!

