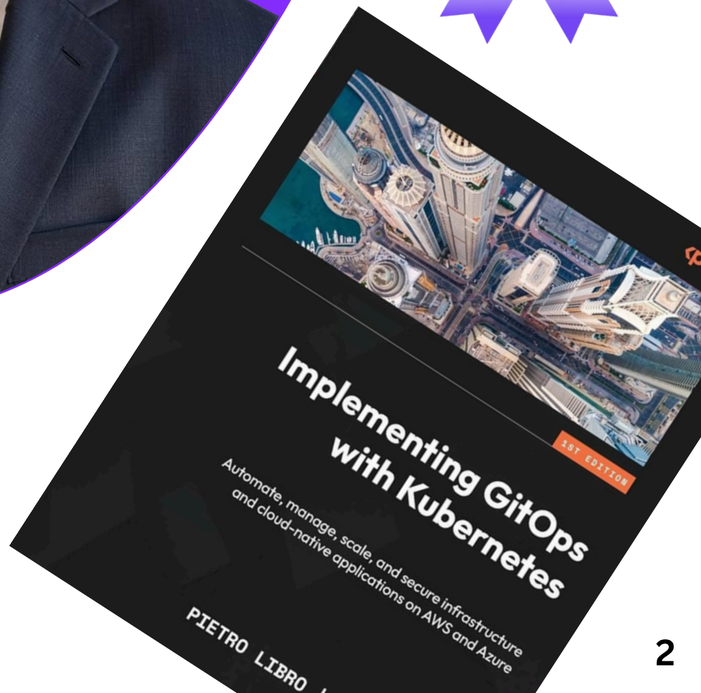
How to build a multi-tenancy Internal Developer Platform with GitOps and vCluster

Artem Lajko

- Platform Engineer and Ambassador
- CNCF Kubestronaut
- Ambassador
- Published Author



Roadmap and Rules

- ~45 minutes presentation
- ~30 minutes live demos
- ~10–15 minutes for discussion / Q&A

- Just One Guideline: Let’s keep questions for before or after the demo.
  - You’re welcome to post them in the chat as we go.
  - We’ll make sure to get to them all!

Target 

- Introduction to Kubernetes and Internal Developer Platforms
- The Role of Platform Engineering in Building and Managing an IDP
- Implementing GitOps with Argo CD and vCluster to Manage Your IDP Seamlessly
- Cost-Efficient Strategies for Multi-Tenant IDPs
- 5 Common Pitfalls When Choosing the Right Solution and How to Avoid Them
- Why Internal Developer Platforms and Platform Engineering matters
- Live Demos 😱

Terminology

- Internal Developer Platform (IDP)
- DevOps Engineer vs Platform Engineer

- Developer

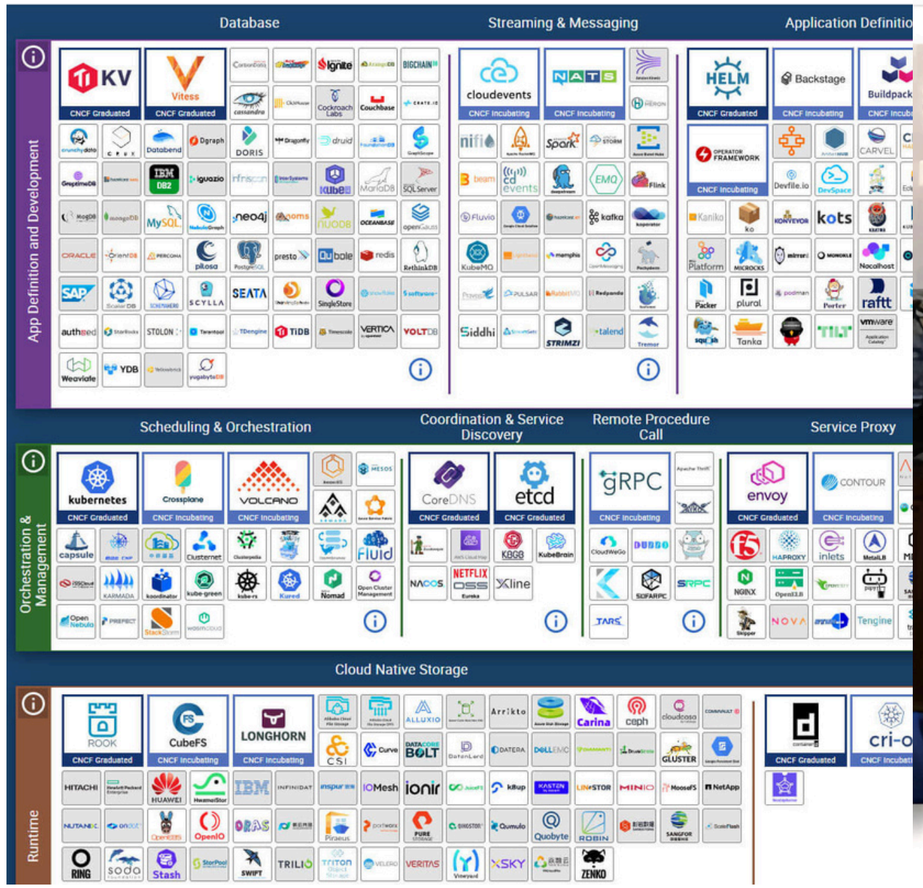
- Developers

- Catalog != Helm Charts
- CI/CD (Push based)
- GitOps (Pull based)

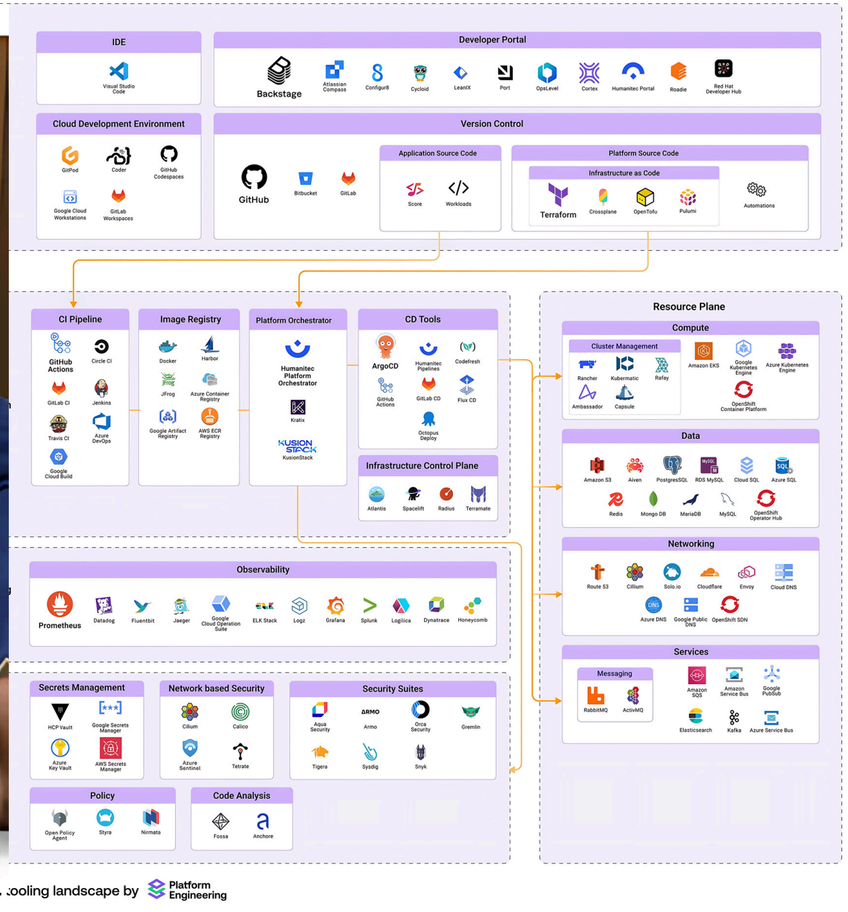
Operator != Orchestrator

Internal Developer Platform

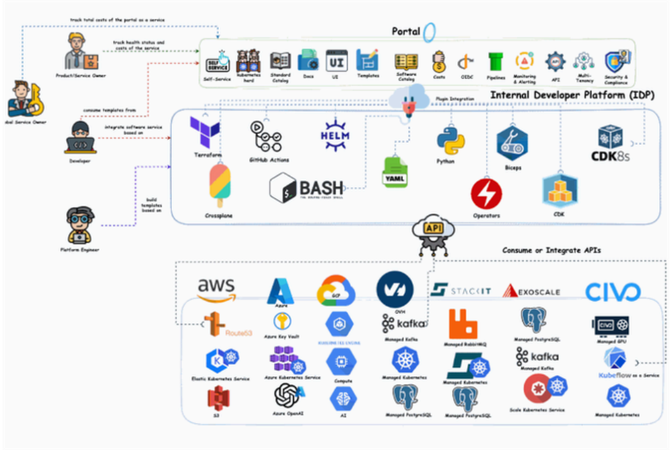
CNCF Landscape



Platform Tooling Landscape



Internal Developer Platforms: A Real Thing or Just a Trend?



Internal Developer Platforms: A Real Thing or Just a Trend?

14 min read · Aug 20, 2024

286 5

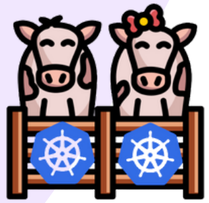
Lifetime

Aug 19, 2024 – Today · Updated every 24 hours

15.6K	8.1K	52%
Views	Reads	Read ratio ⓘ

Internal Developer Platform Requirements

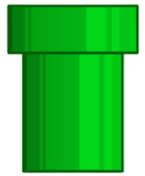
Self-Service



kubernetes herd

Standard Catalog

Docs



Pipelines

Monitoring & Alerting

Software Catalog

OIDC

Multi-Tenancy

Security & Compliance

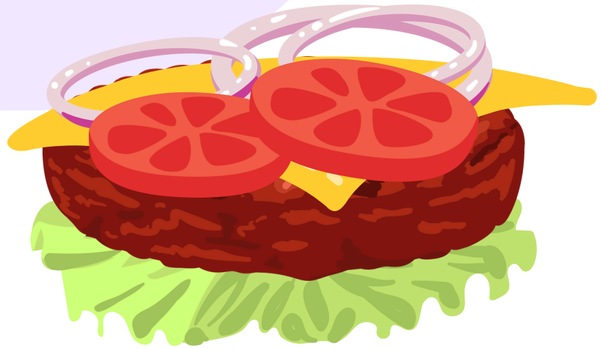
API

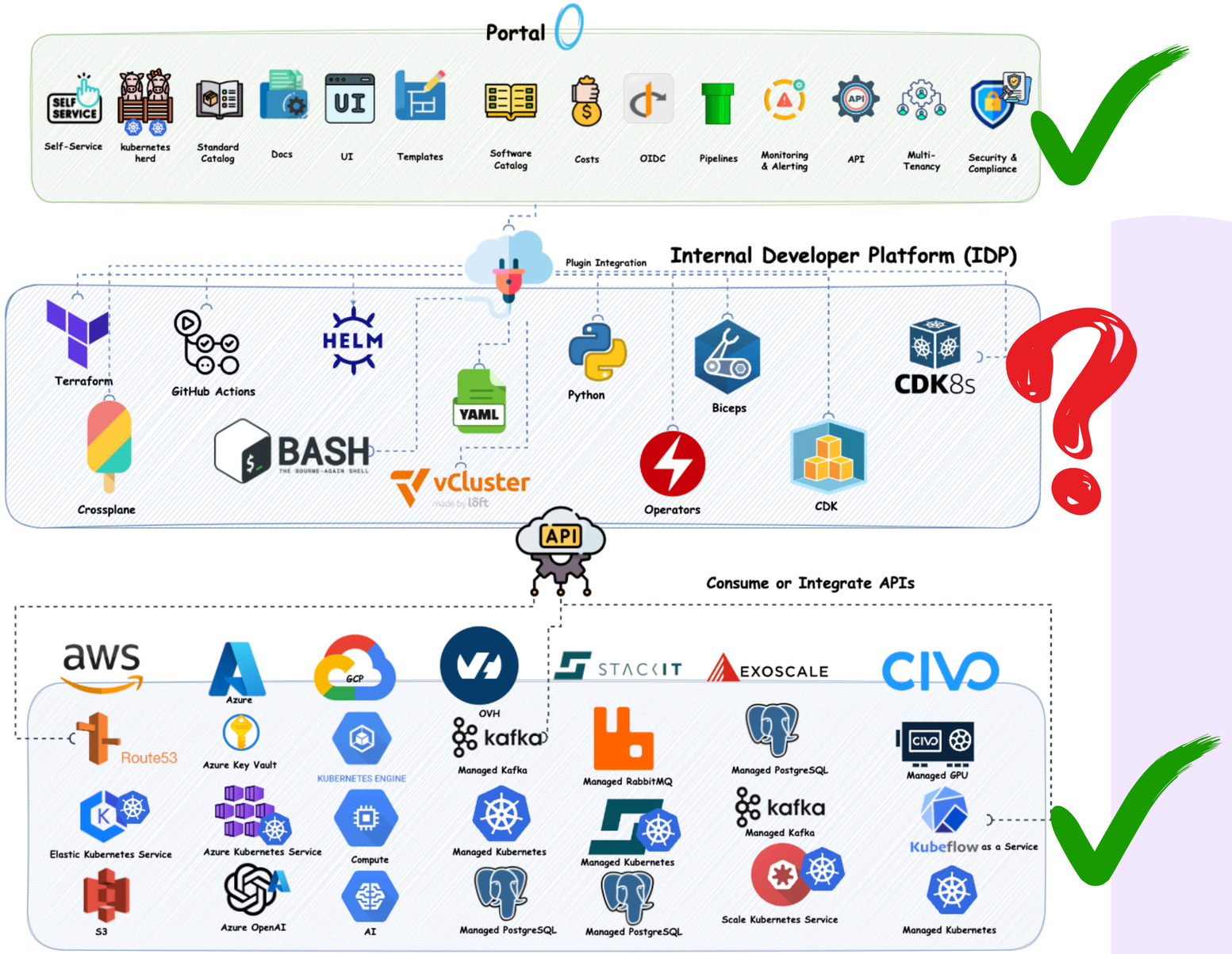
UI

Templates

Costs

Internal Developer Platform: 3 Layer



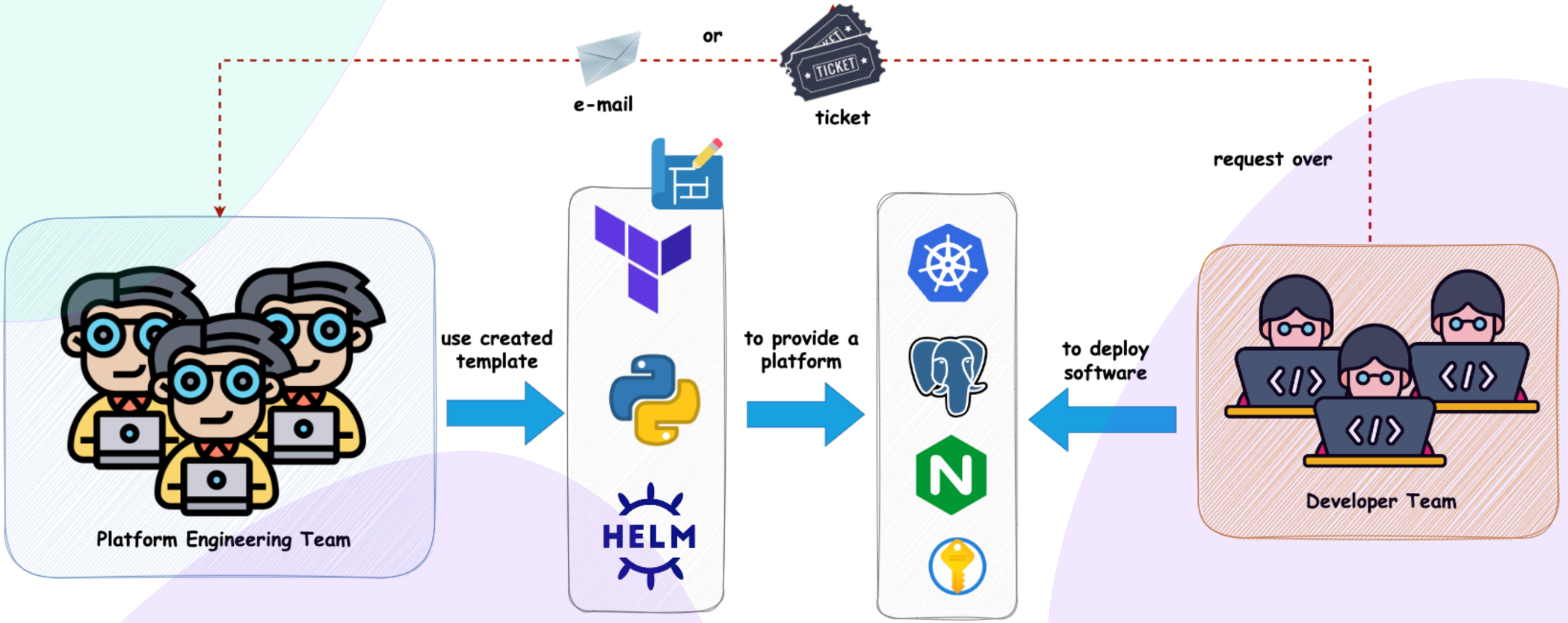


Platform Engineering

Is **ABOUT** **abstracting or compose** individual services, building an overlay, and making them **self-service consumable** by the **end user**.

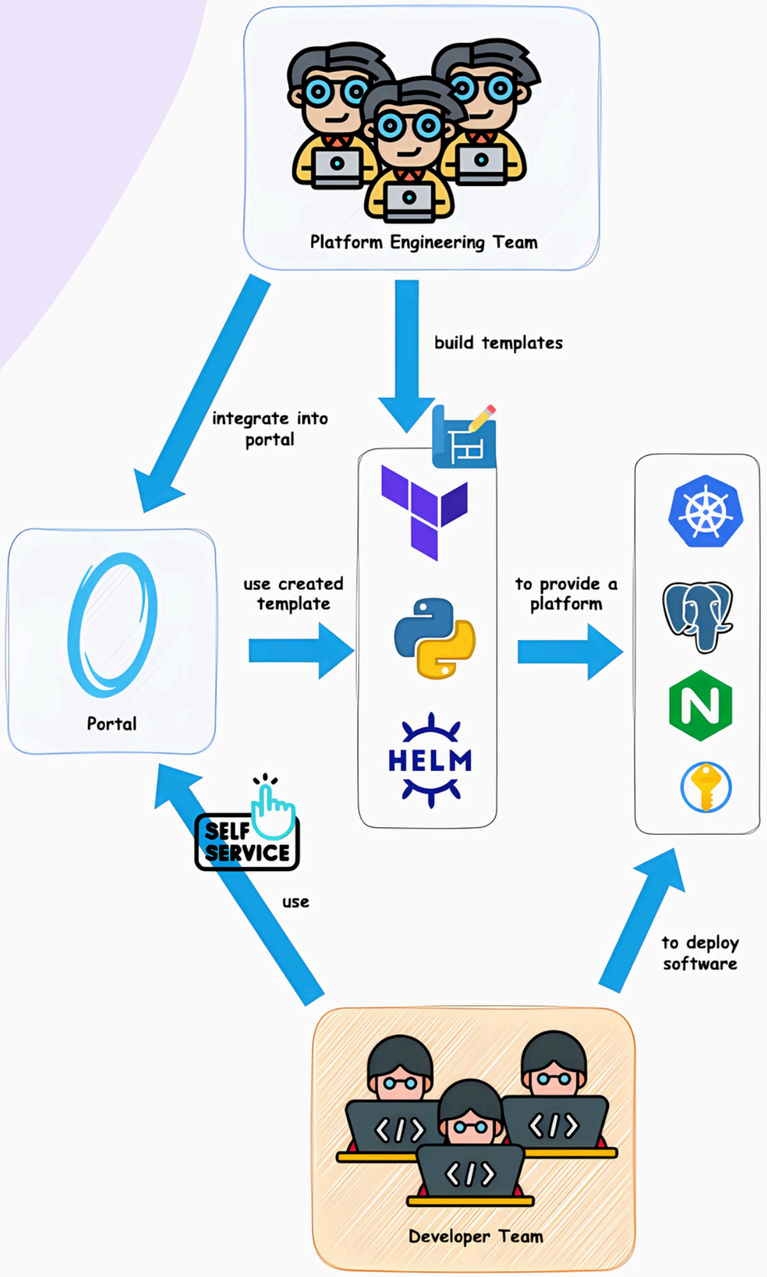
Is **NOT ABOUT** **abstracting or compose** individual services, building an overlay, and using them internally to **provide infrastructure**. If that's your approach, you're **not** a Platform Team—you're a **Managed Infrastructure Team**.

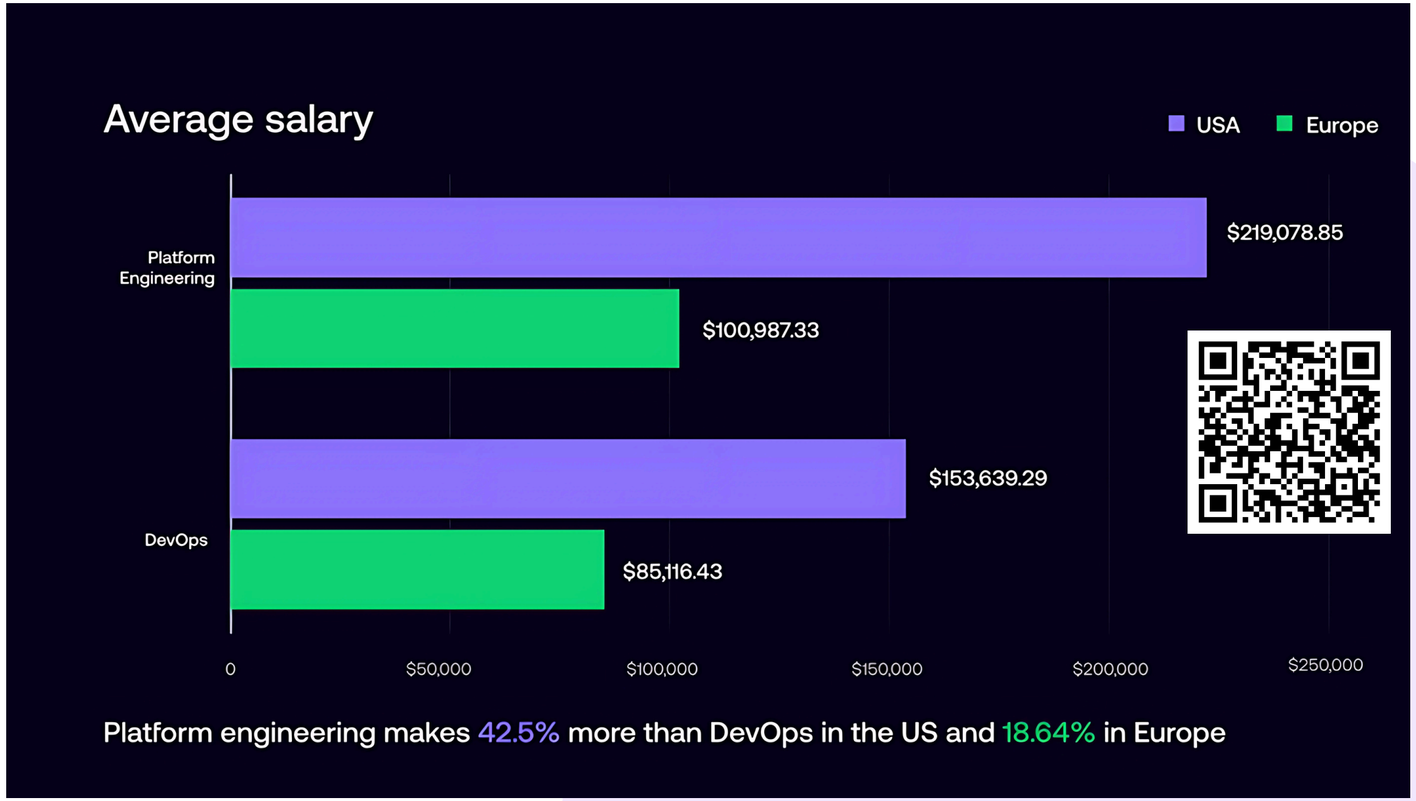
Managed Infrastructure Team



Infra Teams owns the Platform!

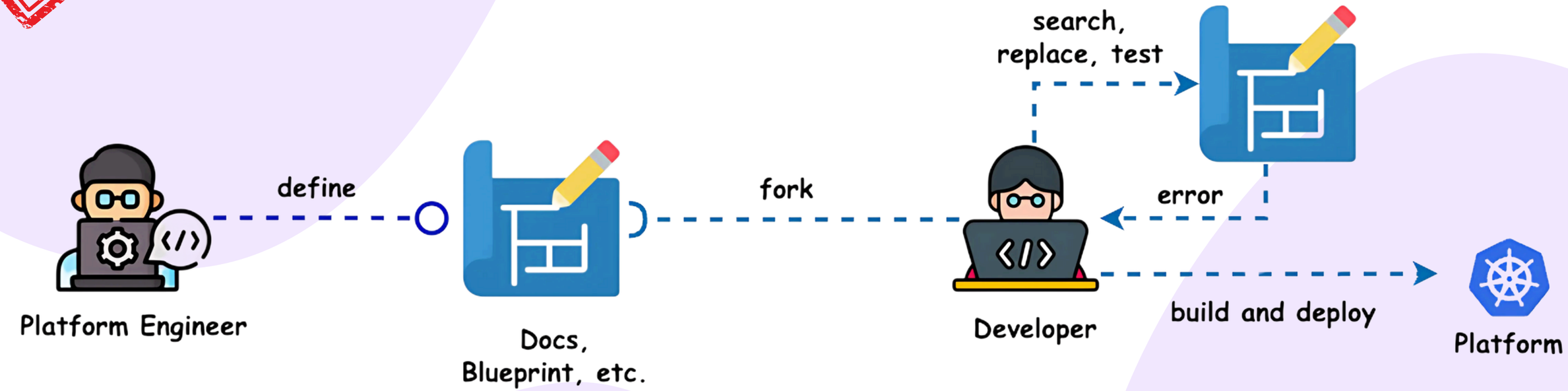
Platform Engineering Team





Developer Teams owns the Platform!

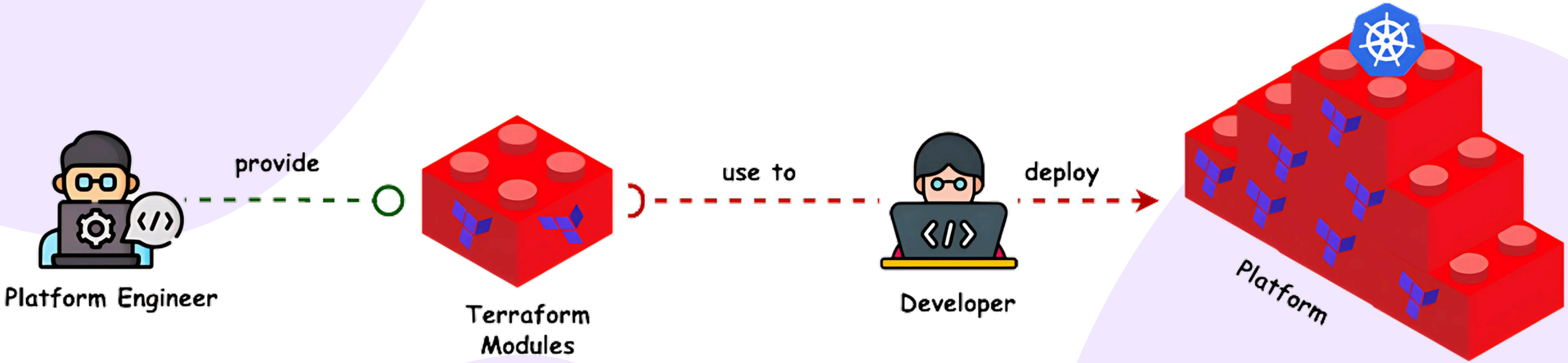
Internal Developer Platform: Blueprint



- Low Automation Levels
- Hard to Scale and Maintain
- Lacking a True Self-Service Model
- Highly Error-Prone
- Unclear or Overlooked Day-2 Operations

REJECTED

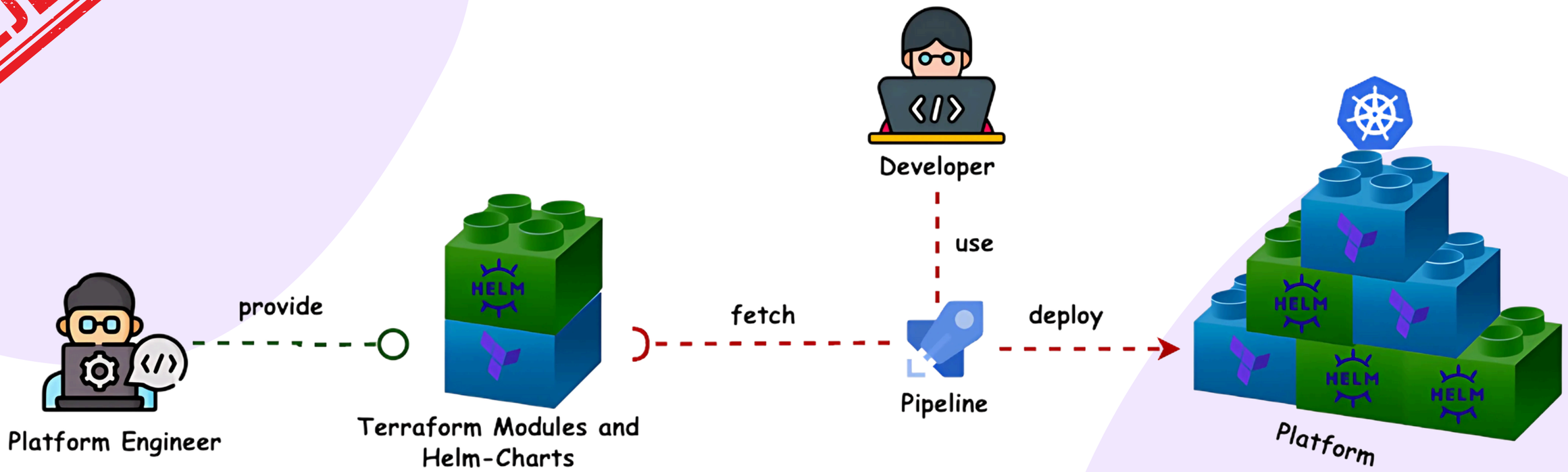
Internal Developer Platform: Bricks (IaC/CaC)



- Limited Automation (Low to Medium Levels)
- Difficult to Scale and Maintain
- No Self-Service Model
- Lower Resilience, Prone to Errors
- Unclear or Insufficient Day-2 Operations

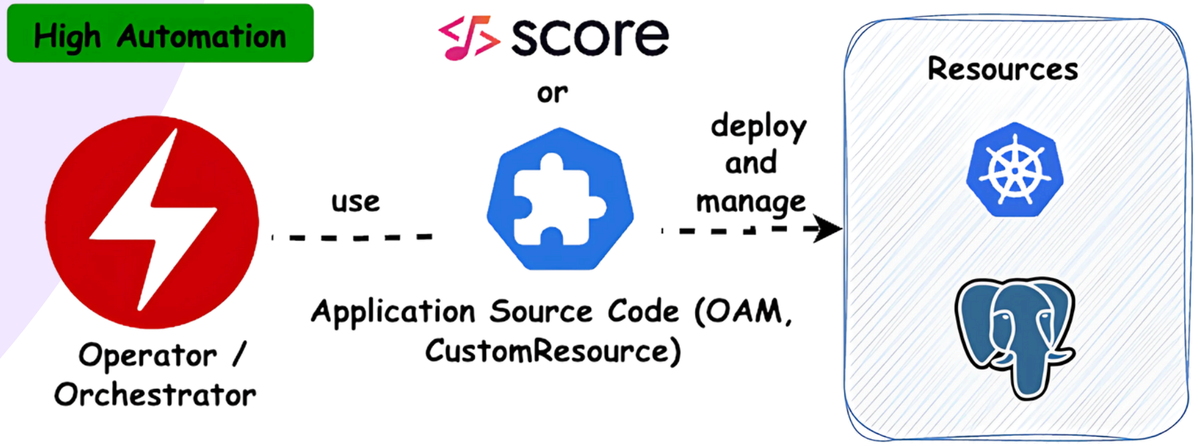
REJECTED

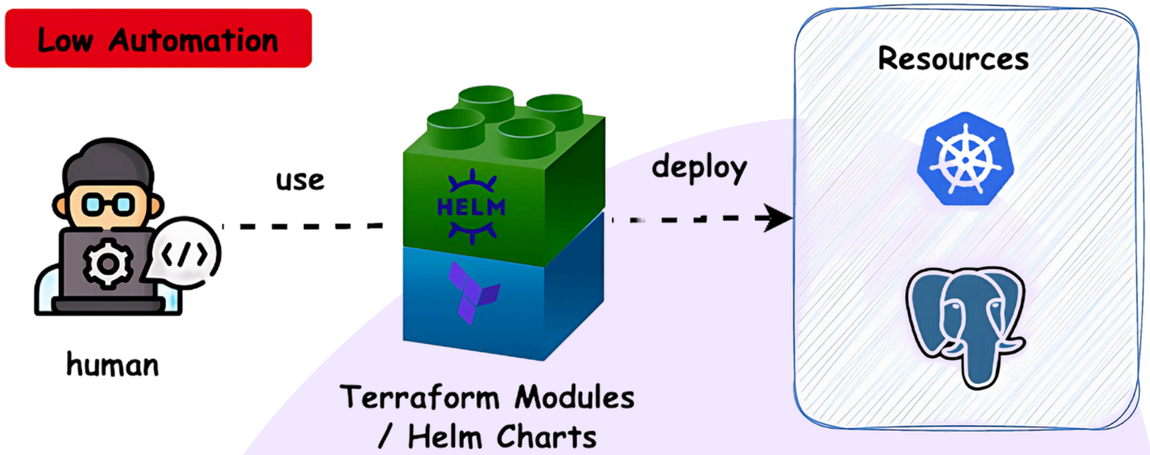
Internal Developer Platform: CI/CD

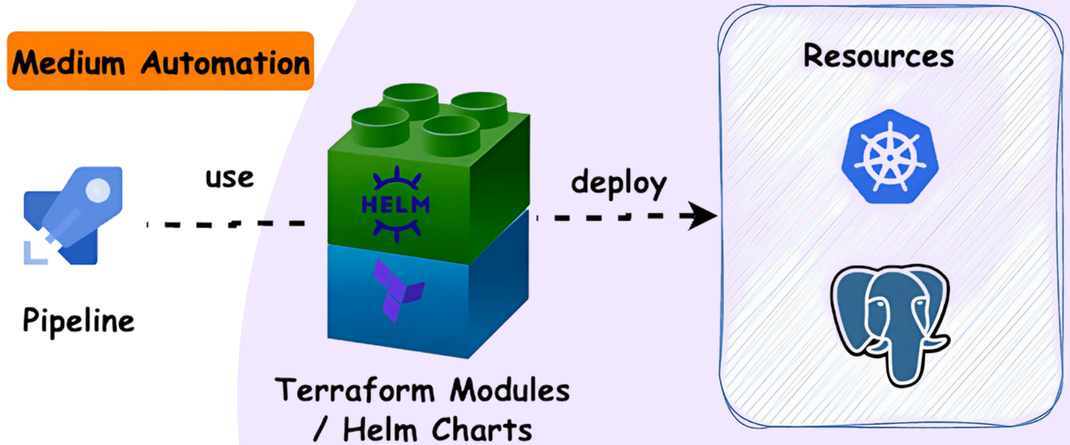


- Moderate Automation Levels
- Improved Scalability and Maintenance, but Still Challenging
- No Self-Service Model
- Reduced, but Persistent Error Risks
- Incomplete or Unclear Day-2 Operations

Degree of Automation and Standardization

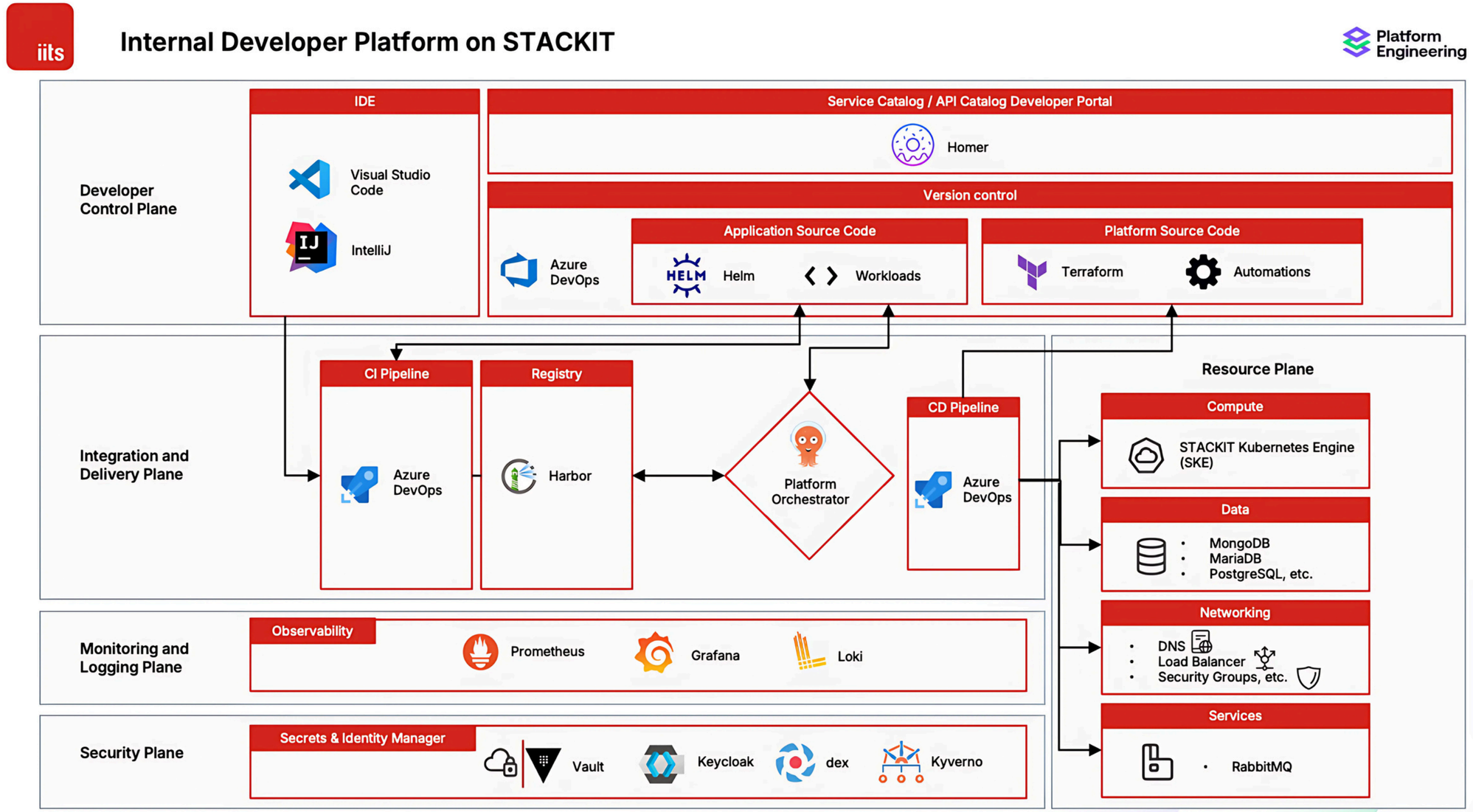






Reference Architecture (High)

Download Slides!



5 Common Pitfalls to Avoid When Building an IDP

1. Ignoring the Education Gap
2. Building on an Unstable Core
3. Just use Backstage
4. Neglecting Day-2 Responsibilities
5. Failing to Measure Value

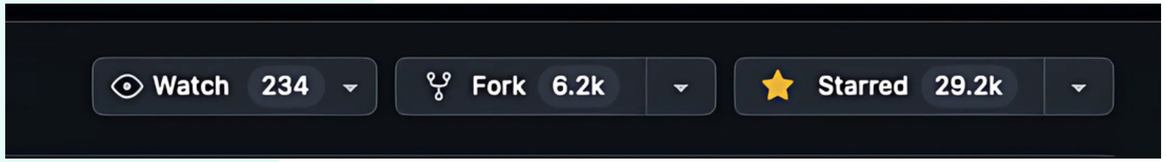
1. Ignoring the Education Gap

- Overlooking the **Education Gap**
- **No Full IDP!** Just Self-Service + UI + Docs
- Lack of **Shared Terminology**
- Spoiler: **IDPs Aren’t Just for Developers**
- Finding the **Right Abstraction Layer**

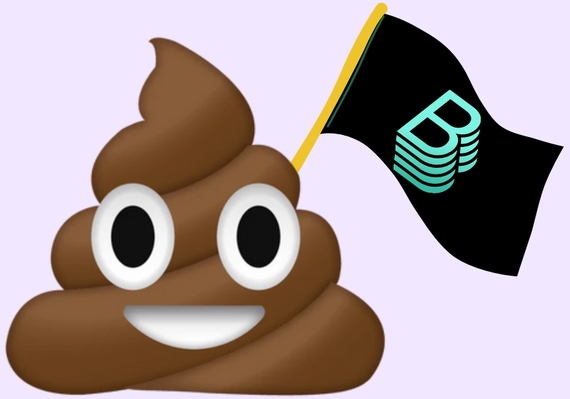
2. Building on an Unstable Core

- **Focusing** on Everything Except **Workflows**
- Adding **Unnecessary Abstraction** Layers
- Increasing **System Complexity**
- Low **Automation Maturity**

3. Just Use Backstage



- Why 'just using Backstage' often ends like this...
  - “Let’s just use **Backstage**—it’s **production-ready**, right?”
  - “Just deploy it, it’s just a **Helm chart**!”
  - “It’ll cover **everything** we need... just check the **docs**!”
  - “Wait, we only get example **Node.js** setups?”
  - “Where’s our **stack**? And who said we’re liking **React**?”

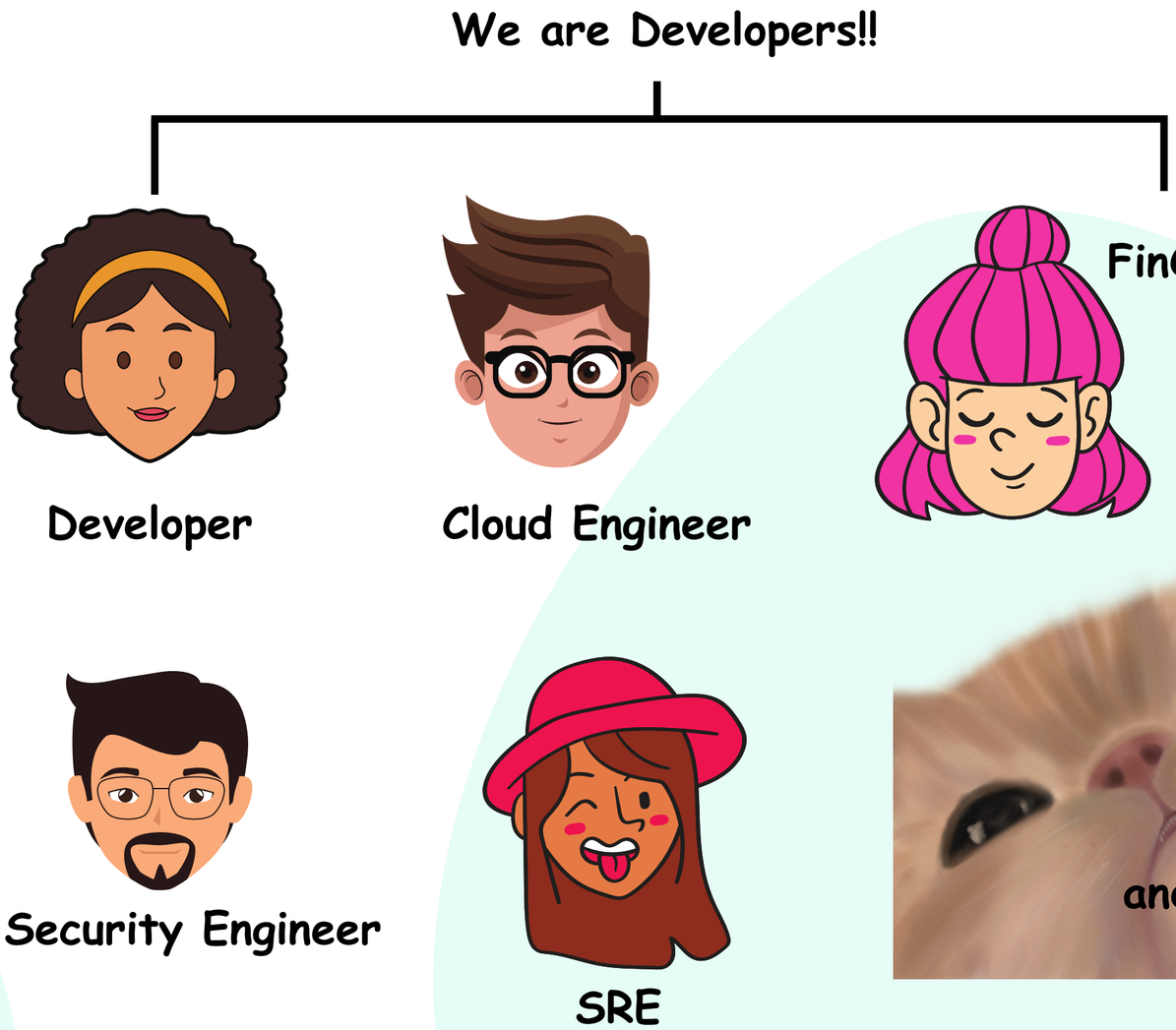


4. Neglecting Day-2 Responsibilities

- Who's Responsible for **Day-2 Operations**?
- Easy, right? **User** of the template, of course.
- But here's a simple example:
  - **PE** provides a template upgrade.
  - **Devs** apply the update, and things break.
  - **Devs** blame **PE**, **PE** blames **Devs**.
- What's really going on here?

5. Failing to Measure Value

Just ask your Developers!



Why IDP matters!

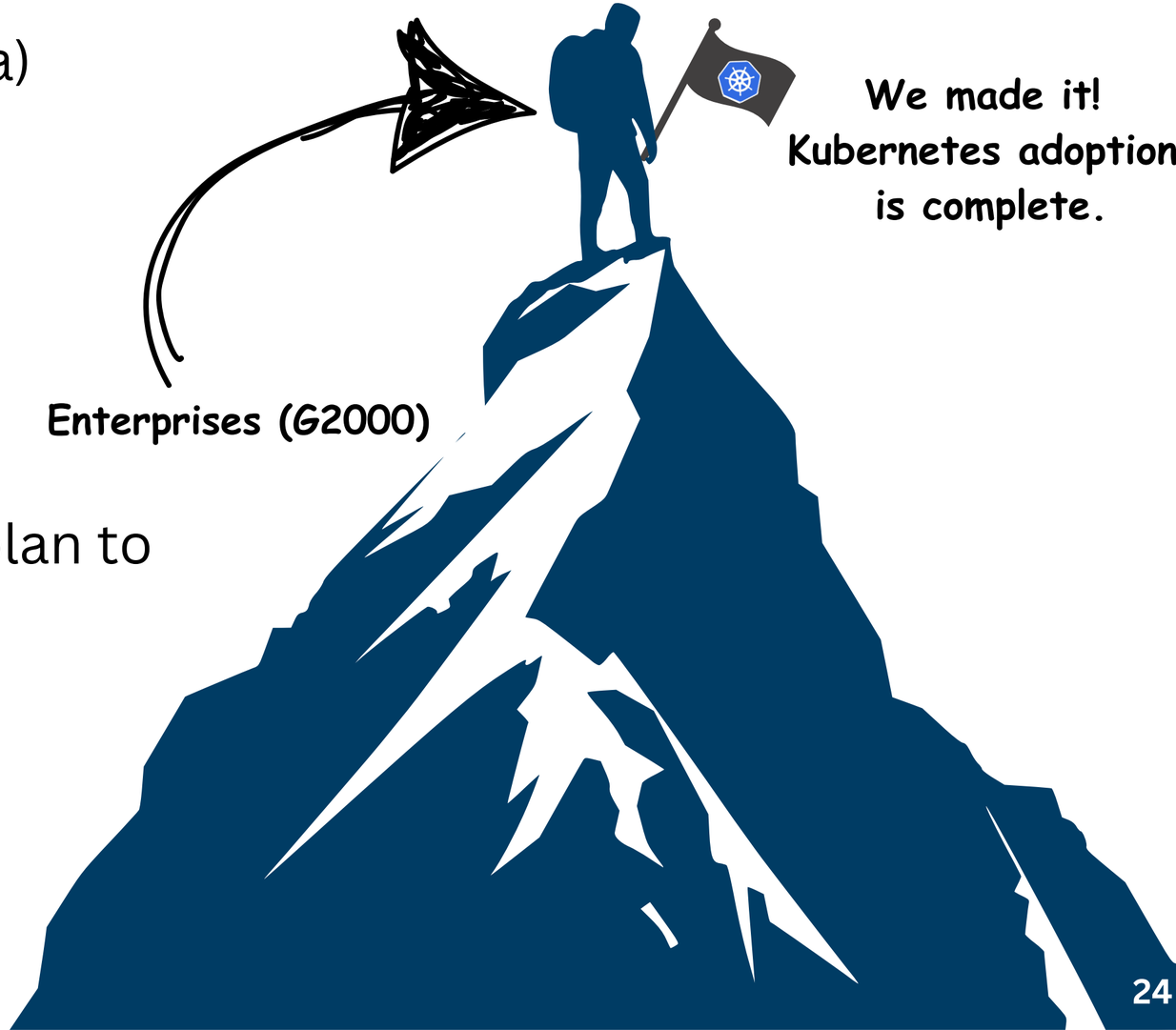
- **Kubernetes** has become the standard for containerization since 2014 (June 6, Joe Beda)

- **By 2027, 90%** of G2000 companies will use container management tools (Gartner)
- In 2023, adoption was below 20% (Gartner)

- **55%** already use platform engineering; **90%** plan to expand it (Google Report)

Gartner

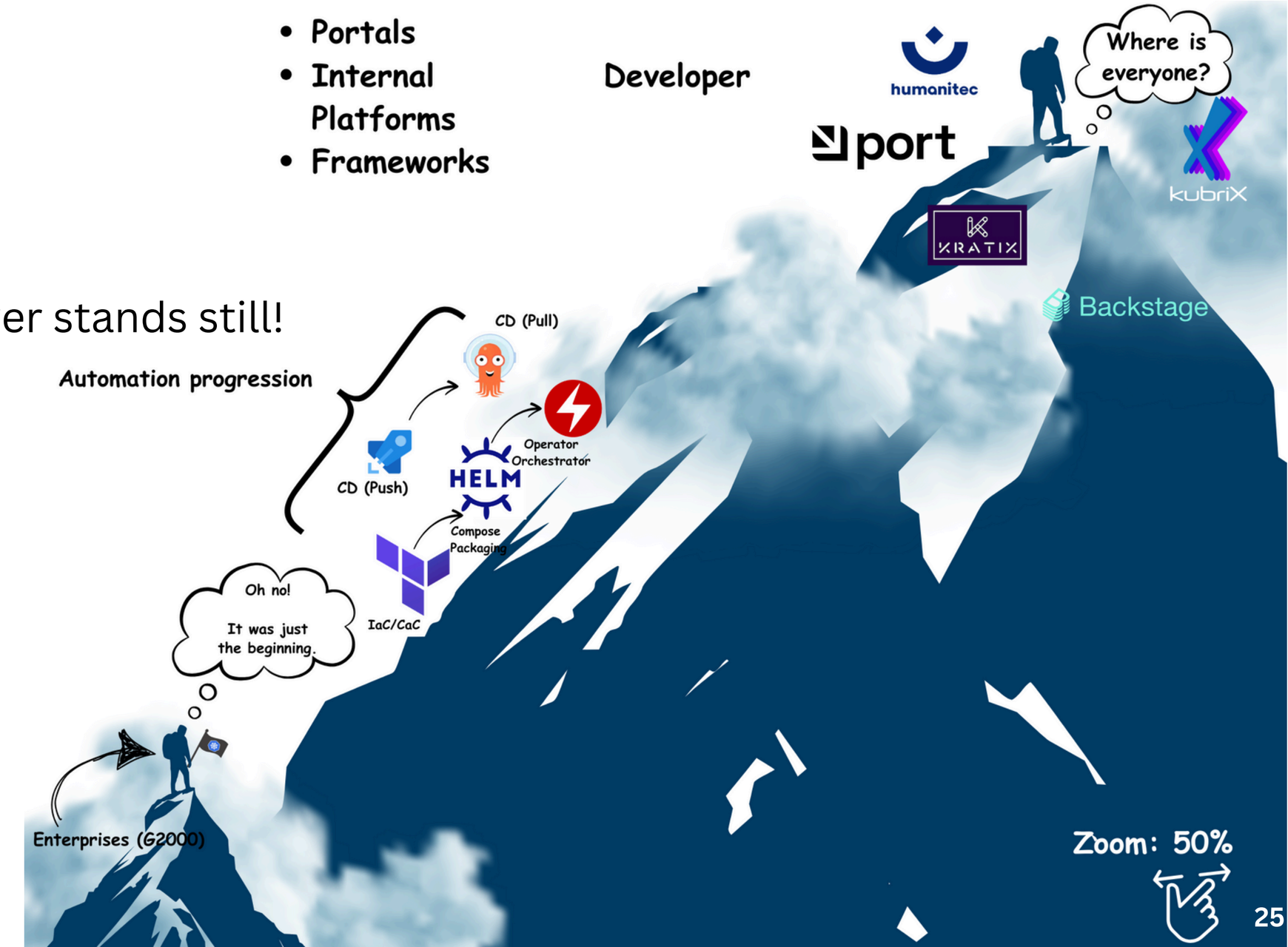
Google Report



Why IDP matters!

Congrats!

But remember—the world never stands still!

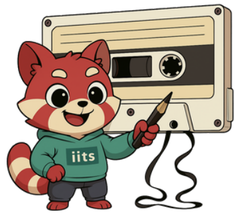


Our Experience & Key Takeaways

- **Prerequisites:** Solid automation level, proper education, and available resources — before getting started!

Team Size	Time-to-Market (TTM)	Exchange Time (ET)
3 PEs, 20 Developers	30–45 minutes	2 days/month
5 PEs, 50 Developers	2–4 hours	7–10 days/month
10 PEs, 400 Developers	1–2 weeks	20 days/month

Recap



- PE **abstracts services** for easy consumption (ideally through self-service) but **don’t abstract a bad core**!
- **Focus on workflows first**—**don’t add extra complexity**. Get things working before scaling up.
- **Boost automation maturity** by selecting the right tools for the right use cases.
- **Close the education gap** before chasing the next big trend.
- **Measure platform impact** by asking users directly—no **need for DORA metrics or KPIs.** Just ask!
- **AI won't save you! (yet)**

Outlook...Q&A?

- **vCluster Webinar:** End of August
- **Book Signing, Talk, and Booth:**
  - ContainerDays, Sept 9–11
- **Book Signing & Giveaway (vCluster booth)**
  - KubeCon NA 2025 Atlanta (November 10-13)
- and more....

LinkedIn

Platform Engineering


don't forget to breathe

Demo Requirements

Why vCluster and SKE?

- **Client Requirements:**
  - **Dev Requests** via Tickets
  - **Sovereign Cloud** (Only europe hyperscaler) -> **STACKIT with SKE**
  - **Cost-Efficient**: Ephemeral Stages, Provisioned Only When Needed
  - **Multi-Tenancy**: Isolated etcd, API, Teams Deploy Different CRD Versions
  - **Developer Experience**: Feels like a Dedicated Cluster

- **Pro Features (not using):**
  - **vCluster & Host-Cluster Hibernate Support**
  - **Sustainability**: Efficient Resource Management
  - **Managed Services**: Argo CD Fully Managed
  - **External Secrets Operator (ESO)**
  - **Isolated Control Plane**

Developer

As a developer, I need a **streamlined GitOps-driven environment** to deploy services quickly with **minimal overhead**, allowing me to work efficiently on my tasks.

Focus on Stable Core

Terraform

GitHub Actions

HELM

YAML

Python

Biceps

CDK8s

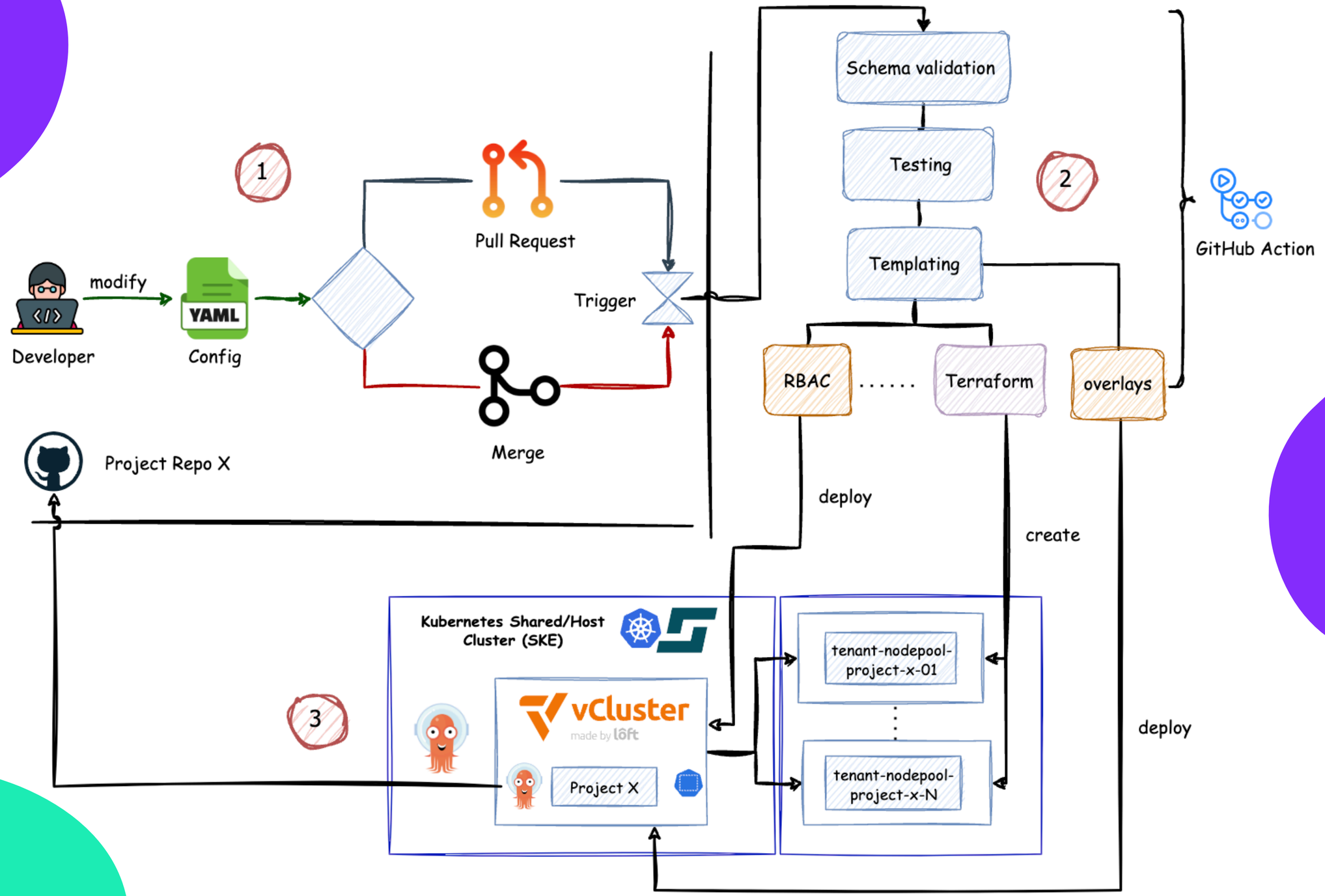
Crossplane

BASH
THE BOURNE-AGAIN SHELL

vCluster
made by loft

Operators

CDK


1
Pull Request
modify
YAML
Developer
Config
Trigger
Merge
Schema validation
Testing
2
Templating
GitHub Action
RBAC
.....
Terraform
overlays
Project Repo X
deploy
create
Kubernetes Shared/Host
Cluster (SKE)
3
vcluster
made by loft
Project X
tenant-nodepool-
project-x-01
tenant-nodepool-
project-x-N
deploy

Demo