

PLATFORM ENGINEERING AND THE FUTURE OF INTERNAL PLATFORM PRODUCTS

Hossein Salahi | Lead Platform Engineer
**Max Körbächer | Founder & Cloud Native
Advisor**

WHERE DEVOPS FAILS?



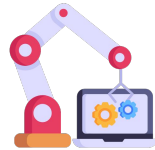
WHERE DEVOPS FAILS?

Core Principles and Promises of DevOps



Collaboration

- Break Down Silos
- Foster a communication Culture
- Cross-Functional Teams
- Organizational-wide engagement



Automation

- Infrastructure as Code
- Continuous Integration
- Continuous Delivery



Continuous Improvements

- Centrally collect and react on events, provide guard rails and intrusion detection



Observability

- Real-time Monitoring
- Both application and infrastructure
- Detect and address proactively



Feedback Loops

- A communication channel between ops and dev
- Analyze success and failure of deployments

WHERE DEVOPS FAILS?

Core Reasons DevOps Initiatives Fail



Lack of vision

Unclear strategic planning and business value

Desired results in terms of time and resources



Unrealistic Expectations

A Faster build, test, and release process

A Secure infrastructure
Setting multi-cloud strategies without understanding the problem



Building DevOps Team

Elimination of bottlenecks
Breaking down silos
But, yet another silo!

Failing to apply it organization wide



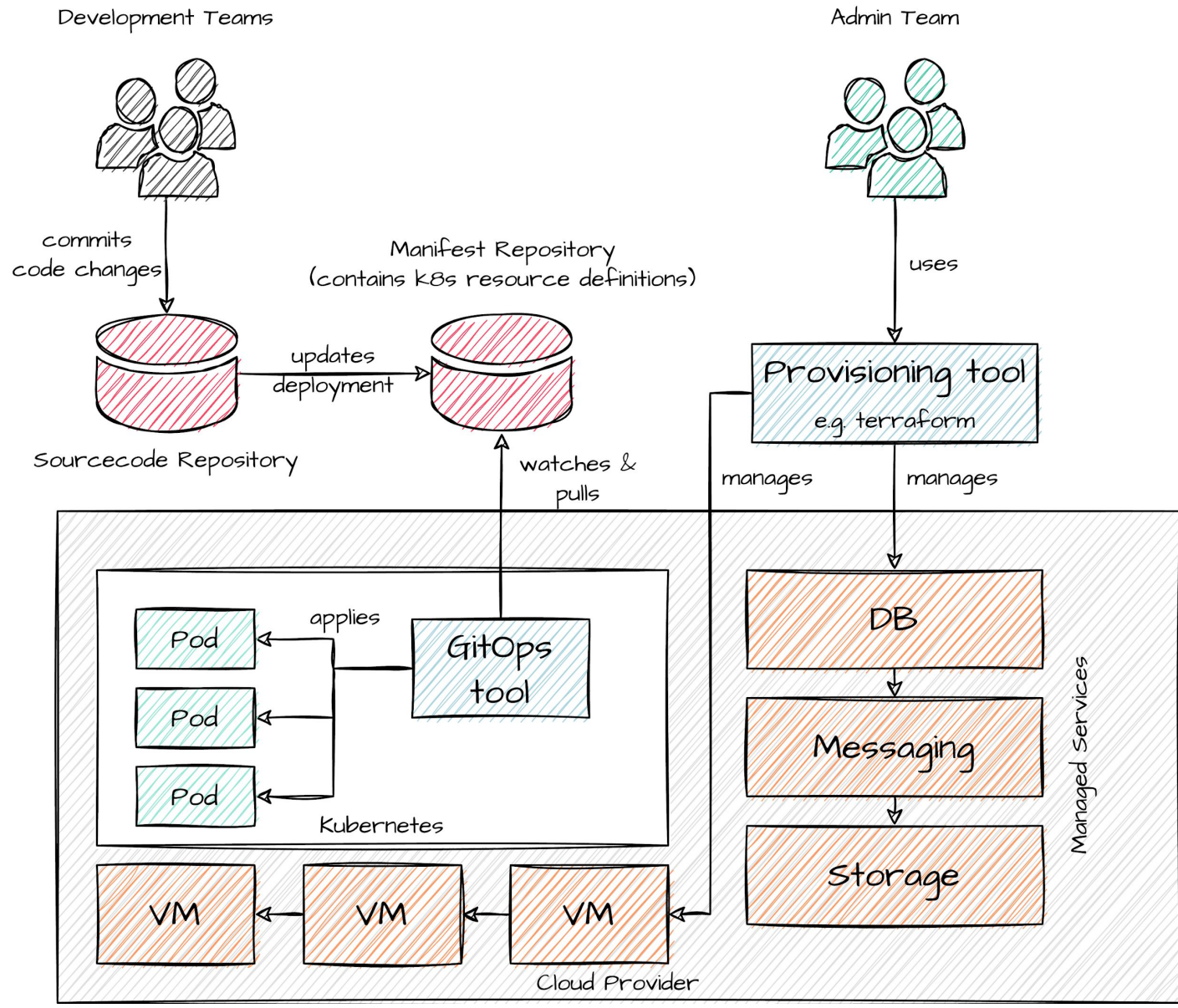
Adaption Approaches

A cultural shift
A mindset rather than tools, speed, or applications



Maintaining Old Structures

Cultural pushback
Building a hybrid structure
Keeping ops and dev in their silos

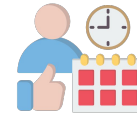


WHY DEVOPS FAILS?

Why may development teams not adopt it?

- They are probably slow
- Not everybody dream to take end-to-end responsibility!
- Too many new tools and they are solving problems
- They were not part of service conceptualization
- Service discovery and ownership
- Container platform is not the best abstraction layer

PLATFORM ENGINEERING Definition



A discipline of designing and building toolchains and workflows



A layer abstraction to simplify underlying infrastructure provisioning and management



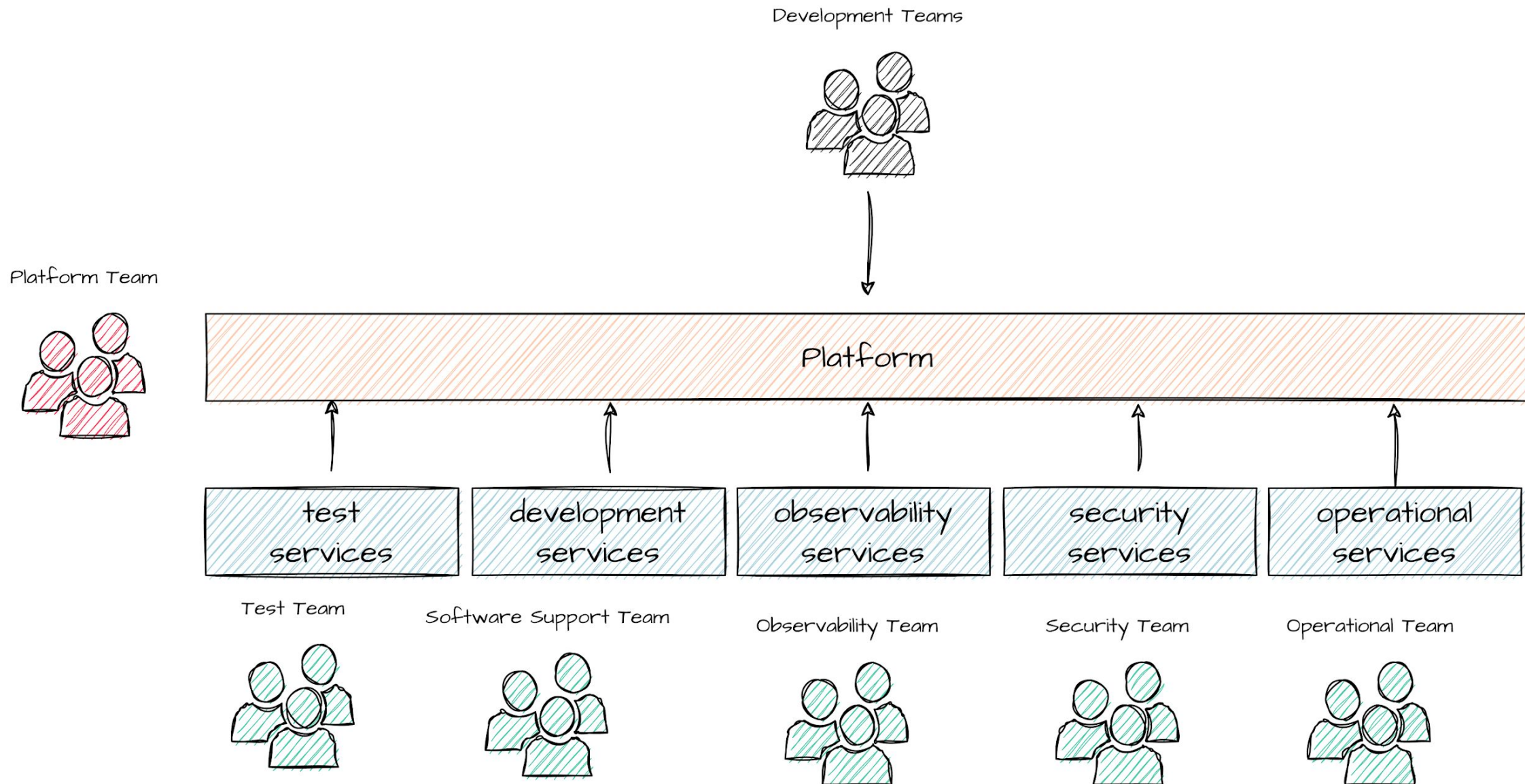
A standardized, automated, scalable environments

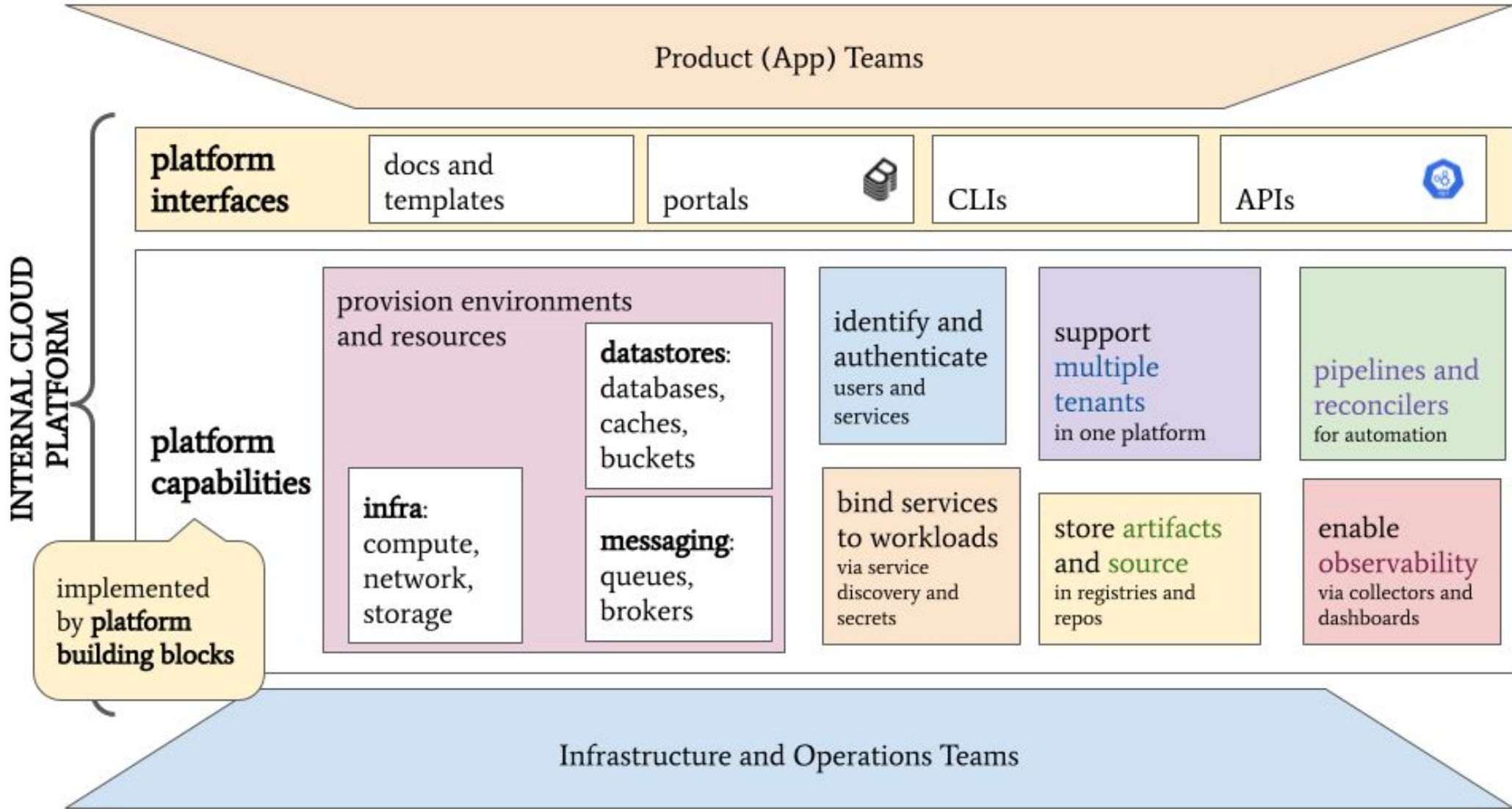


Self-service capabilities for development teams



More efficient application delivery





PLATFORM ENGINEERING RESPONSIBILITIES

Infrastructure Automation

Standardization

Security and Compliance

Scalability and Resilience

Workflow Orchestration

Self-Service Capabilities

Observability

Collaboration and Feedback

CRAFT YOUR OWN PLATFORMS



**“BEFORE YOU CAN BUILD A
PLATFORM YOU NEED TO KNOW WHAT
WILL BE ITS PURPOSE”**

PLATFORM LAYERS

Consist of multiple components that interact in different speed with each other



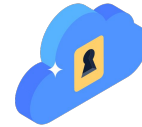
Developer Control Plane

Consist of any Dev (Software / Platform) required tools for coding & versioning



Integration & Delivery Plane

Build, test & deploy software and components



Security & Compliance Plane

Centrally collect and react on events, provide guard rails and intrusion detection



Observability & Operability Plane

Tooling provided to operate the platform, and in some cases also the application



Resource Plane

Usually a cloud provider, IaaS or hypervisor

Capability Plane



Features which the platform provides for the users in lives “within” the platform

IDP REFERENCE

Developer
Control Plane

IDE



VSCode



GitHub
Codespace



Gitpod

Service Catalog / API Catalog Developer Portal



Backstage

Version control



GitHub

Application Source Code



Buildpacks



Workloads

Platform Source Code

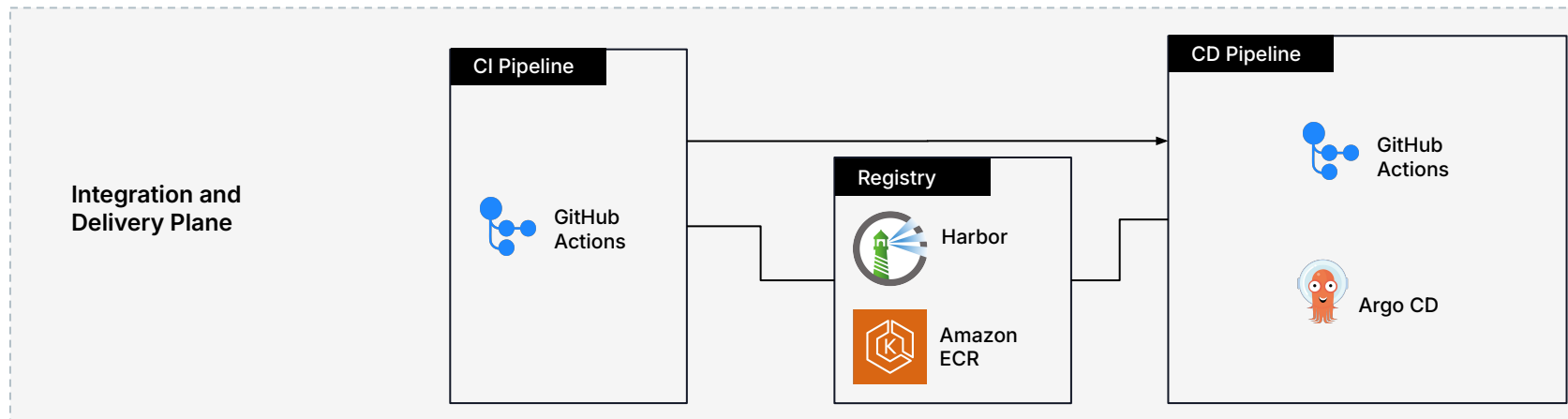


Terraform



Automations

IDP REFERENCE



IDP REFERENCE

Observability &
Operations
Plane

Observability



Prometheus



Grafana



Zipkin



Dynatrace

IDP REFERENCE

Security &
Compliance
Plane

Secrets & Identity Manager



HCP Vault

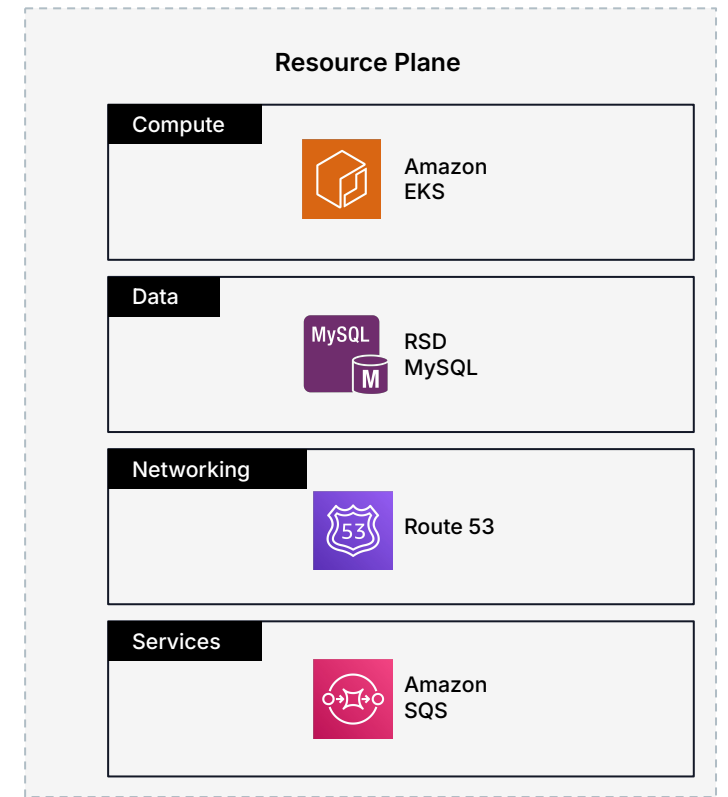


Keycloak

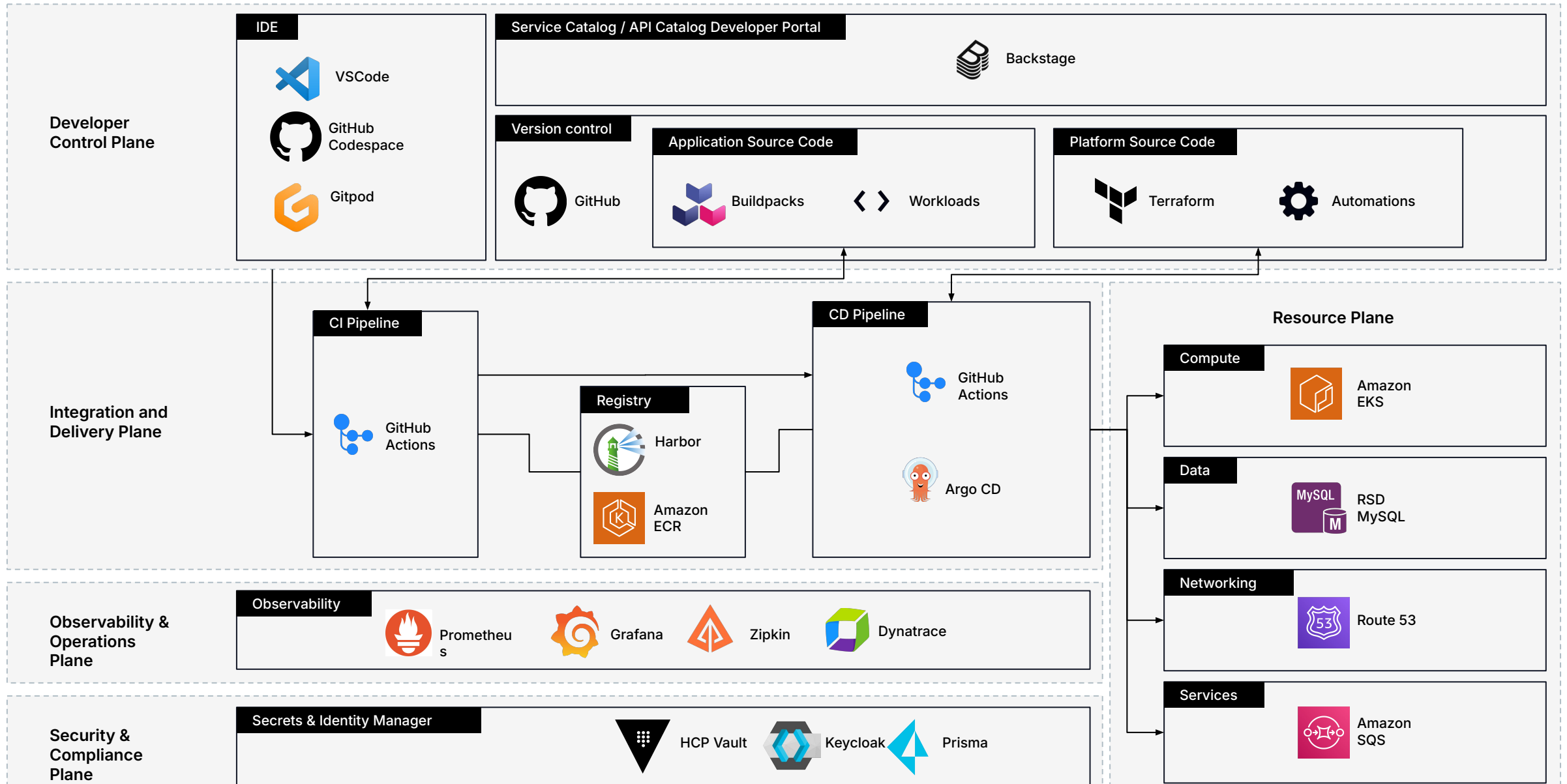


Prisma

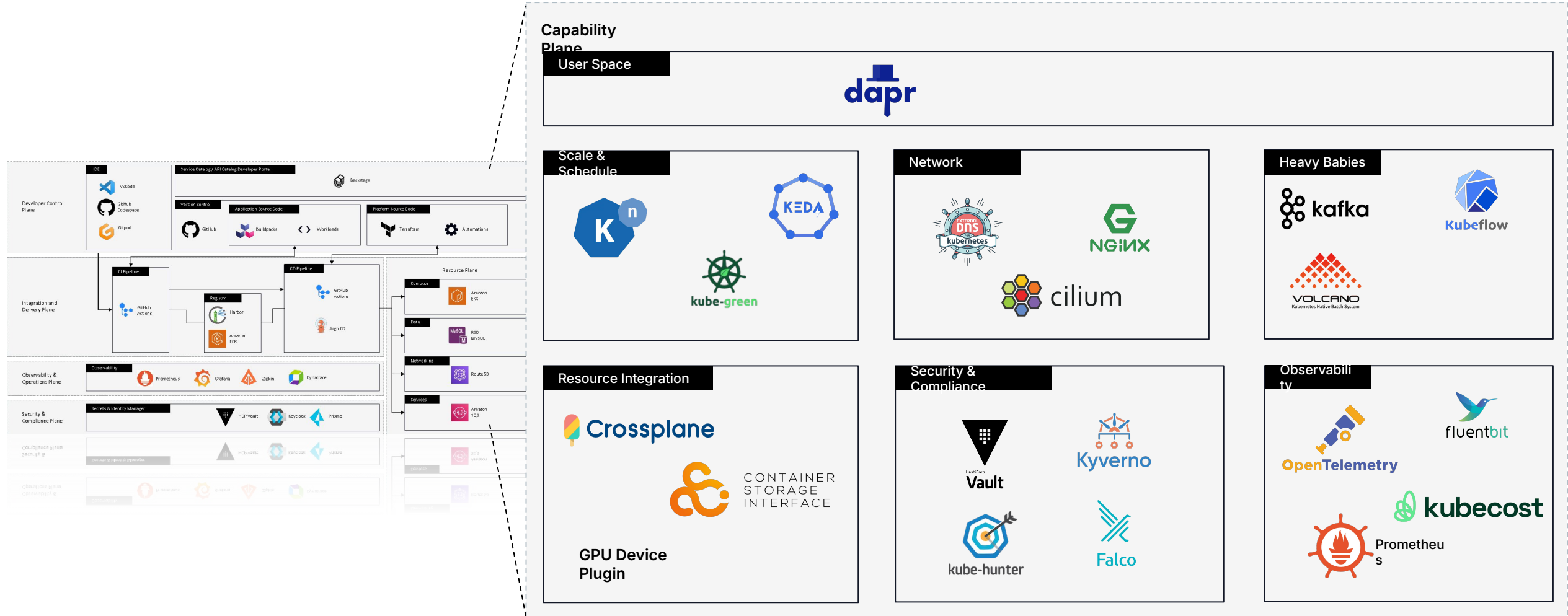
IDP REFERENCE



IDP REFERENCE



IDP REFERENCE



Capability Plane

User Space



Scale & Schedule



Network



Heavy Babies



Resource Integration



GPU Device
Plugin



CONTAINER
STORAGE
INTERFACE

Security & Compliance



Observability

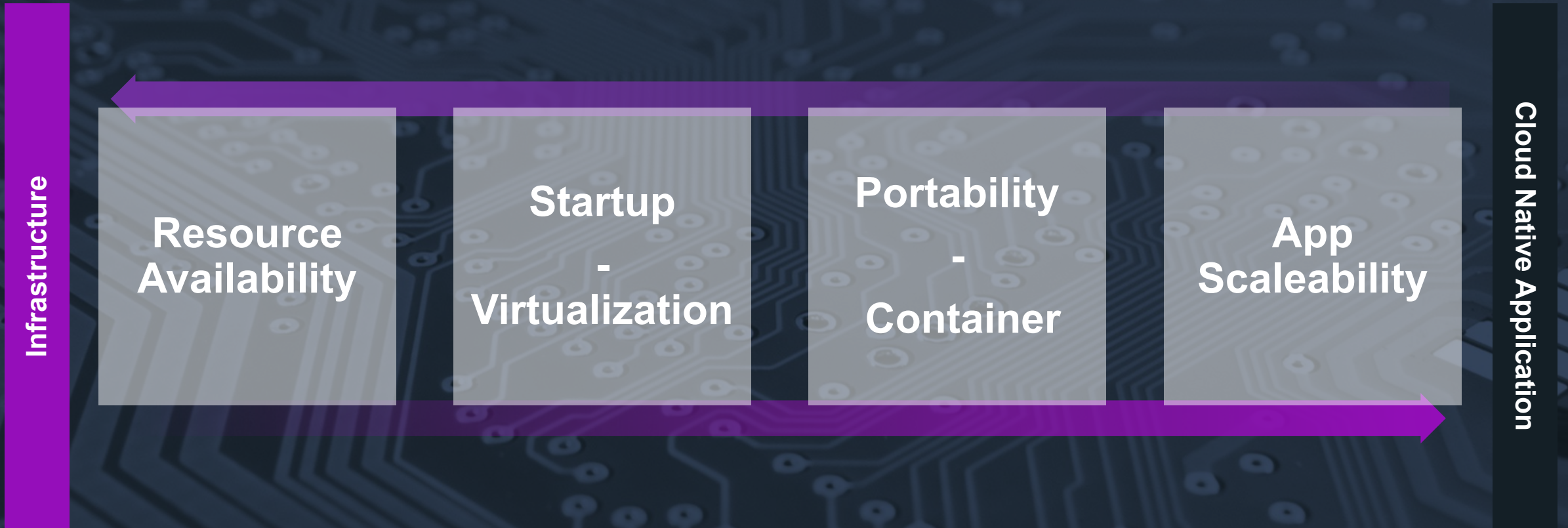


Prometheu
s



INFLUENCING

Hardware and software effects each other



INFLUENCING

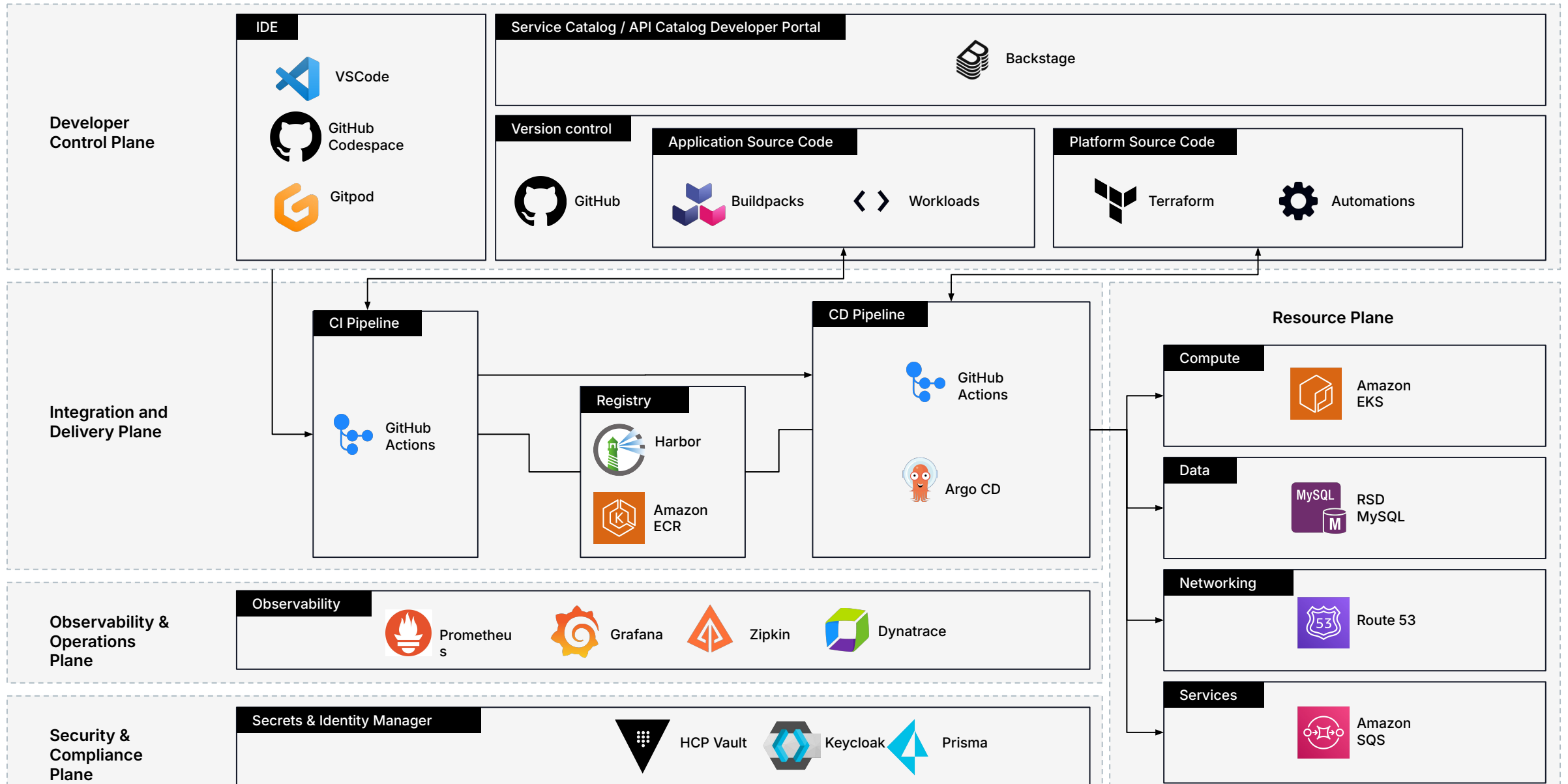
Hardware and software effects each other

Infrastructure

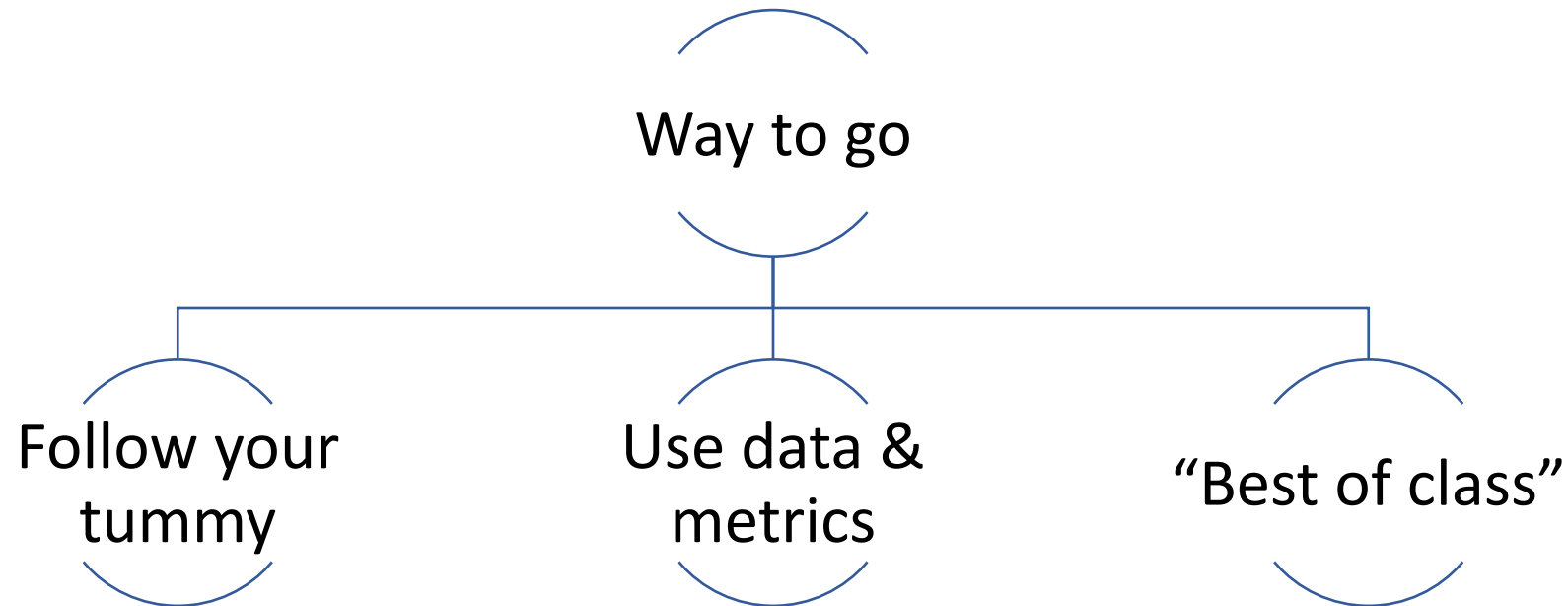
Cloud Native Application

**PLATFORMS NEEDS TO PROVIDE THE
CAPABILITIES TO ADJUST TO ANY
WORKLOAD WITHOUT HARMING THE
UNDERLAYING INFRASTRUCTURE**

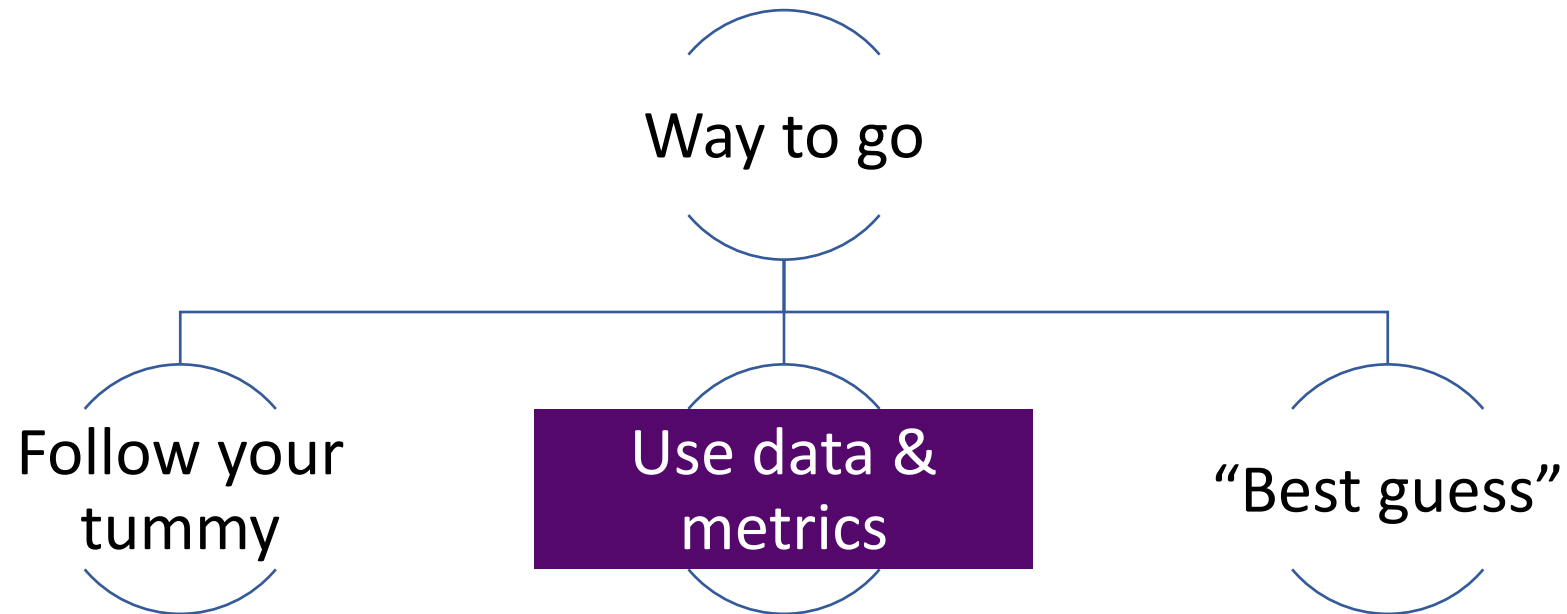
IDP REFERENCE



HOW TO FIND THE RIGHT APPROACH?



HOW TO FIND THE RIGHT APPROACH?



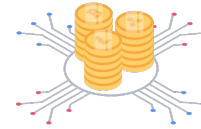
GOLDEN PATH KEY INDICATORS



Frequency of deployments in %



Development time incl. waiting time & error/bug fixing in h



Operational effort in h



Alignment and communication effort in h



Failure rate in %

DEFINE GOLDEN PATH

Collect data on certain observation points

- Add/Update
 - Services
 - Resources
 - Configurations
- Change architecture
- Create new environments
- Onboarding new developers
- Roll back of failed deployments
- Debugging & error tracing
- Blocked environments
- Waiting for other teams

DATA EXAMPLE TO FIND GOLDEN DATA

Dev Steps	Frequency %	Dev Time h	Ops Effort h	Alignment/ Comms h	Failure Rate %
Add/Update Services	40%	16	8	2	30%
Add/Update Resources	20%	8	24	3	6%
Add/Update Configurations	60%	1	1	0	10%
Change architecture	3%	60	20	10	60%
Create new environments	4%	24	24	3	25%
Onboarding new developers	20%	80	16	4	80%
Roll back of failed deployments	17%	10	20	2	90%
Debugging & error tracing	45%	10	16	4	5%
Blocked environments	8%	16	1	2	4%
Waiting for other teams	16%	16	36	5	30%

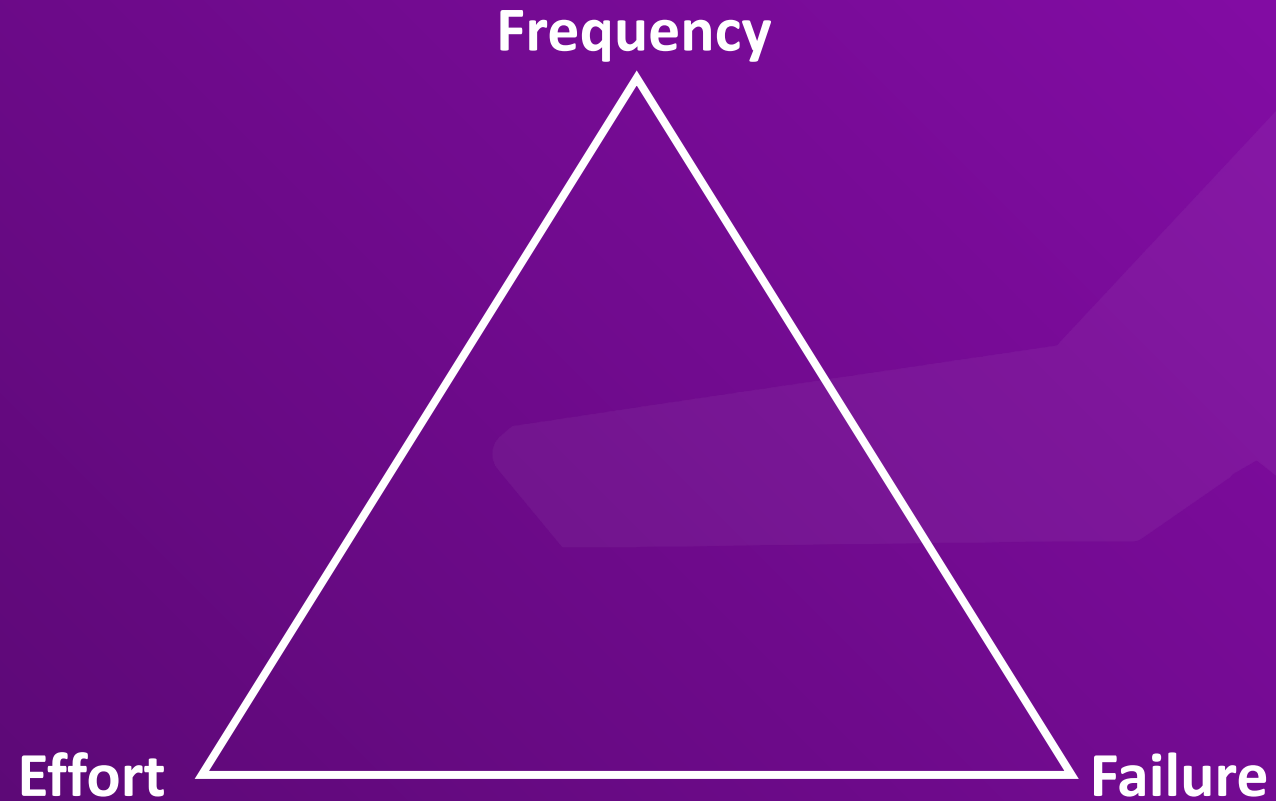
DATA EXAMPLE TO FIND GOLDEN DATA

Dev Steps	Frequency %	Dev Time h	Ops Effort h	Alignment/ Comms h	Failure Rate %
Add/Update Services	40%	16	8	2	30%
Add/Update Resources	20%	8	24	3	6%
Add/Update Configurations	60%	1	1	0	10%
Change architecture	3%	60	20	10	60%
Create new environments	4%	24	24	3	25%
Onboarding new developers	20%	80	16	4	80%
Roll back of failed deployments	17%	10	20	2	90%
Debugging & error tracing	45%	10	16	4	5%
Blocked environments	8%	16	1	2	4%
Waiting for other teams	16%	16	36	5	30%

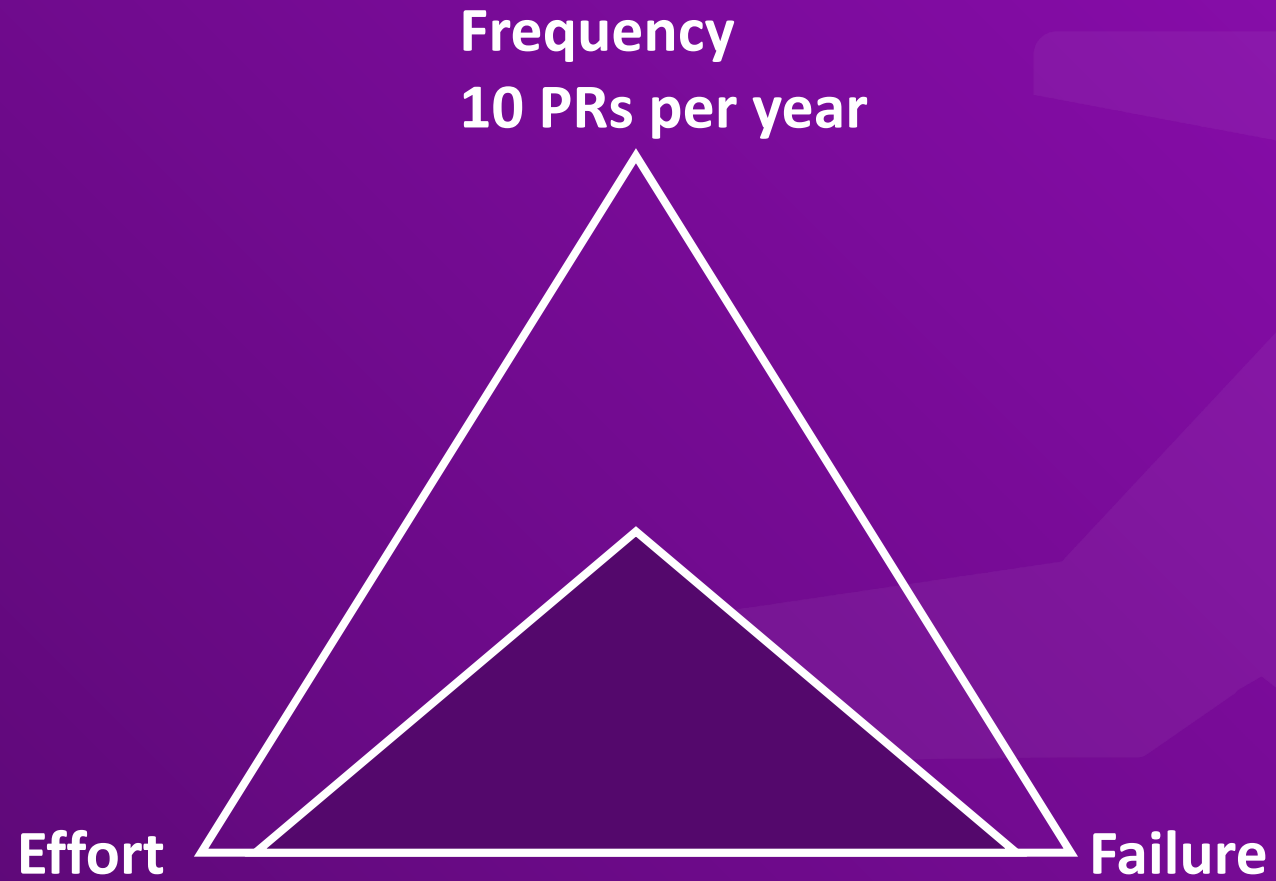
Worst 3 per KPI

Best 3 per KPI

IDENTIFYING THE STARTING POINT



CASE 1



DATA EXAMPLE TO FIND GOLDEN PATH

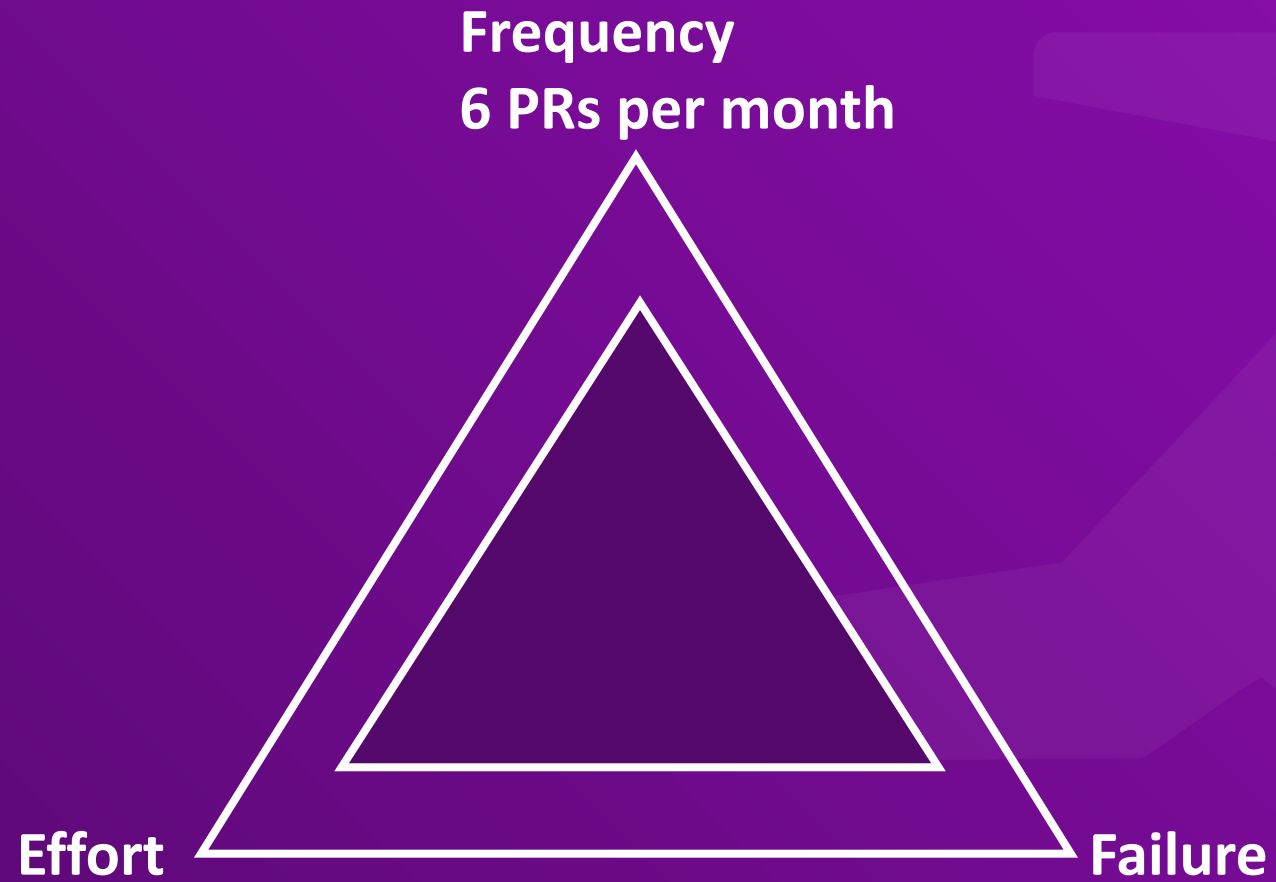
Dev Steps	Frequency %	Dev Time h	Ops Effort h	Alignment/ Comms h	Failure Rate %
Add/Update Services	40%	16	8	2	30%
Add/Update Resources	20%	8	24	3	6%
Add/Update Configurations	60%	1	1	0	10%
Change architecture	3%	60	20	10	60%
Create new environments	4%	24	24	3	25%
Onboarding new developers	20%	80	16	4	80%
Roll back of failed deployments	17%	10	20	2	90%
Debugging & error tracing	45%	10	16	4	5%
Blocked environments	8%	16	1	2	4%
Waiting for other teams	16%	16	36	5	30%

Worst 3 per KPI

Best 3 per KPI

Golden Path

CASE 2



DATA EXAMPLE TO FIND GOLDEN PATH

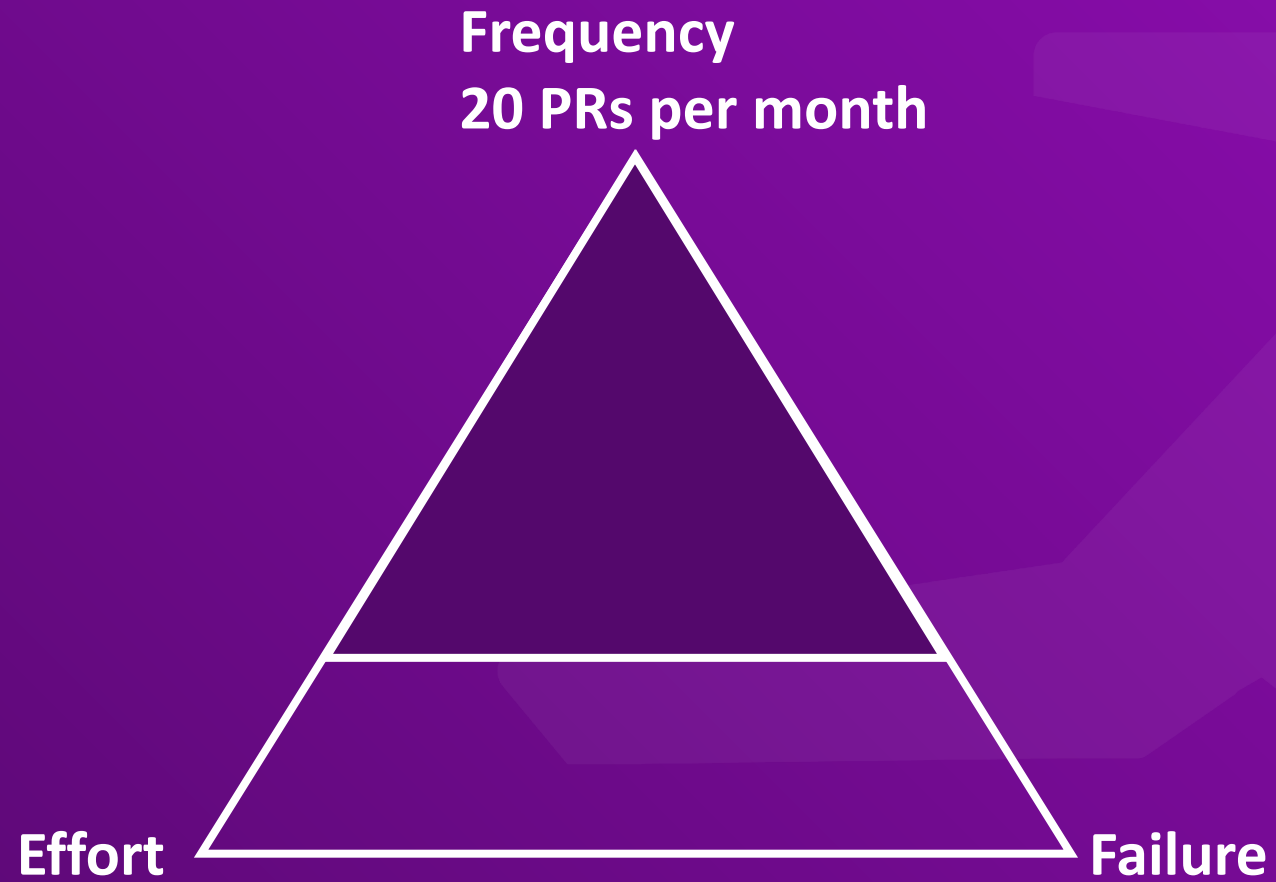
Dev Steps	Frequency %	Dev Time h	Ops Effort h	Alignment/ Comms h	Failure Rate %
Add/Update Services	40%	16	8	2	30%
Add/Update Resources	20%	8	24	3	6%
Add/Update Configurations	60%	1	1	0	10%
Change architecture	3%	60	20	10	60%
Create new environments	4%	24	24	3	25%
Onboarding new developers	20%	80	16	4	80%
Roll back of failed deployments	17%	10	20	2	90%
Debugging & error tracing	45%	10	16	4	5%
Blocked environments	8%	16	1	2	4%
Waiting for other teams	16%	16	36	5	30%

Worst 3 per KPI

Best 3 per KPI

Golden Path

CASE 3



DATA EXAMPLE TO FIND GOLDEN PATH

Dev Steps	Frequency %	Dev Time h	Ops Effort h	Alignment/ Comms h	Failure Rate %
Add/Update Services	40%	16	8	2	30%
Add/Update Resources	20%	8	24	3	6%
Add/Update Configurations	60%	1	1	0	10%
Change architecture	3%	60	20	10	60%
Create new environments	4%	24	24	3	25%
Onboarding new developers	20%	80	16	4	80%
Roll back of failed deployments	17%	10	20	2	90%
Debugging & error tracing	45%	10	16	4	5%
Blocked environments	8%	16	1	2	4%
Waiting for other teams	16%	16	36	5	30%

Worst 3 per KPI

Best 3 per KPI

Golden Path

RECAPITULATE THE STEPS

1. Define its primary purpose

- What will it be good for?
- How does it support your org?

2. Evaluate your current situation (facts & figures)

- Collect the data, don't let people guess, take at least 2-4 weeks to measure it

3. Decide for the golden path

- Define which KPI is your primary driver
- Think of the paredo optimum

4. Prioritize implementations

- Develop a plan which feature should come first, clarify dependencies and ensure it will not have a negative effect on other KPIs

COMMUNITIES & RESPONSIBILITIES



PLATFORMS LIVE BY TWO VALUES

Platforms as a Product not as a Project

Community

Platform Teams have to learn that they also haven't eaten StackOverflow with a golden spoon.

Open your platforms for contributions, open discussions and 2nd round of input.

Product Owner can gain input from many sources.

Responsibilities

Platforms require a long term responsibility and accountability -> become a product.

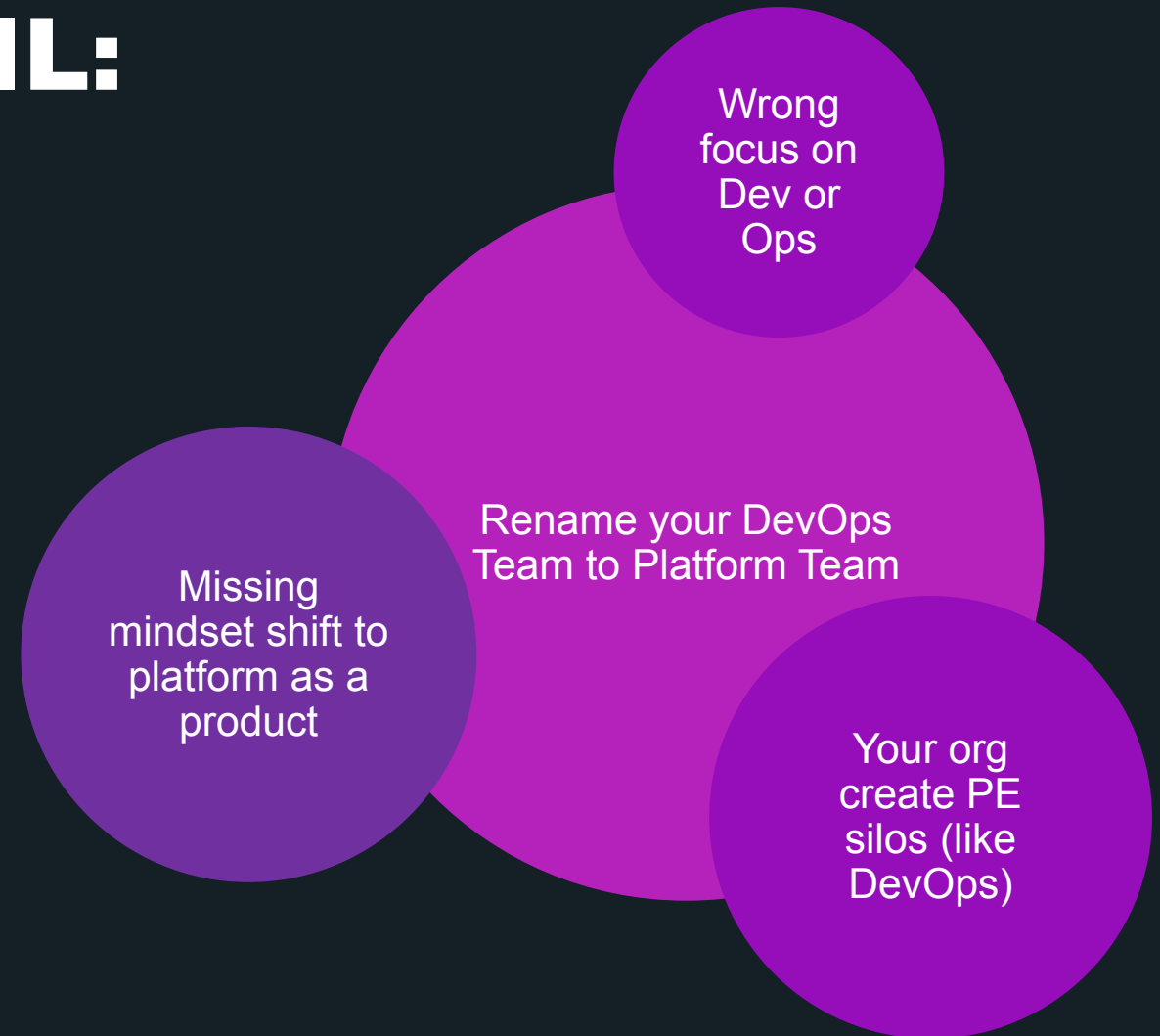
Those responsibilities needs to be clearly defined, to be able to act as thought leader.

But, responsibilities can be shared.

ELSE, YOU WILL FAIL:

Technical “Play Ground”

- Replace old technologies with new (alpha/beta) for no reason or “just be early adopter”
- Which leads often to a setup that is “special”
- Or build portals and features literally no one asked for
- This usually causes hidden and sunken costs



INTERNAL DEVELOPER PLATFORMS



“An Internal Developer Platform (IDP) is a specialized environment or set of tools and services designed to streamline and enhance the software development process within an organization.”

IDP CORE ELEMENTS



APPLICATION CONFIGURATION MANAGEMENT

Scope, Versioning, Portability, and Secret Management

INFRASTRUCTURE ORCHESTRATION

IaC, CI/CD, DNS, Clusters, and other resources

ENVIRONMENT MANAGEMENT

Self-serve fully provisioned environments on demand

DEPLOYMENT MANAGEMENT

Continuous Deployment (CD)

ROLE-BASED ACCESS CONTROL

Manage access on a granular level

WHY AN IDP IS IMPORTANT?

STANDIZATION

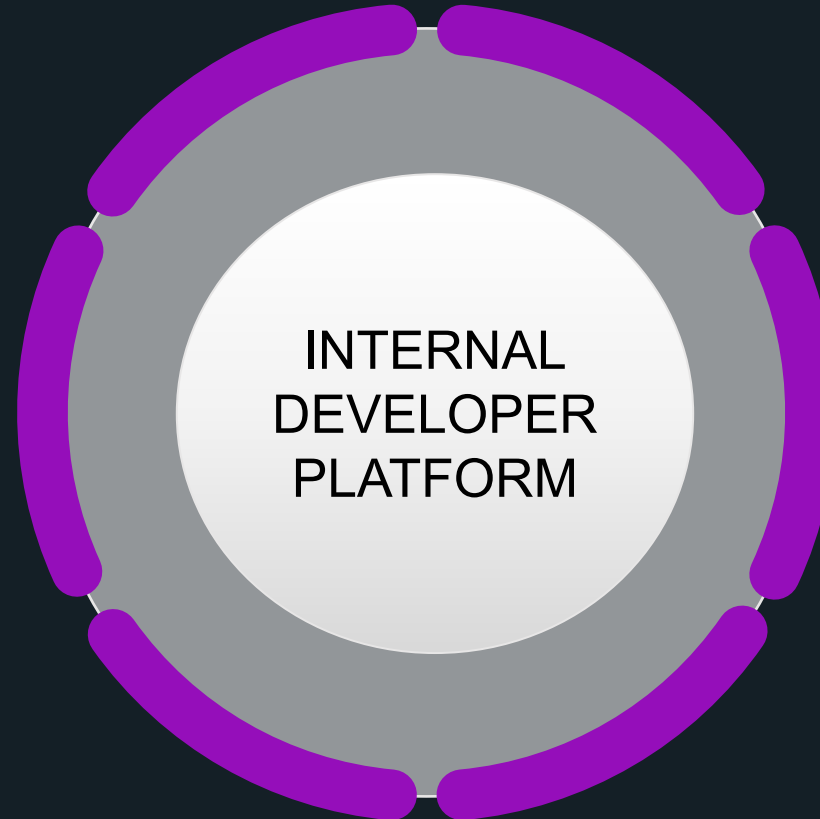
- A standard set of tools and services
- Reduces risk of inconsistencies and errors

PRODUCTIVITY

- Automates setting up and managing development environments building pipelines, and application delivery

GOVERNANCE

- A framework that enables adherence to best practices that complies with security and compliance



COLLABORATION

- A shared platform to collaborate between teams such operations and security

SELF-SERVICE

- Reduce new dev team on boarding time and operational complexities

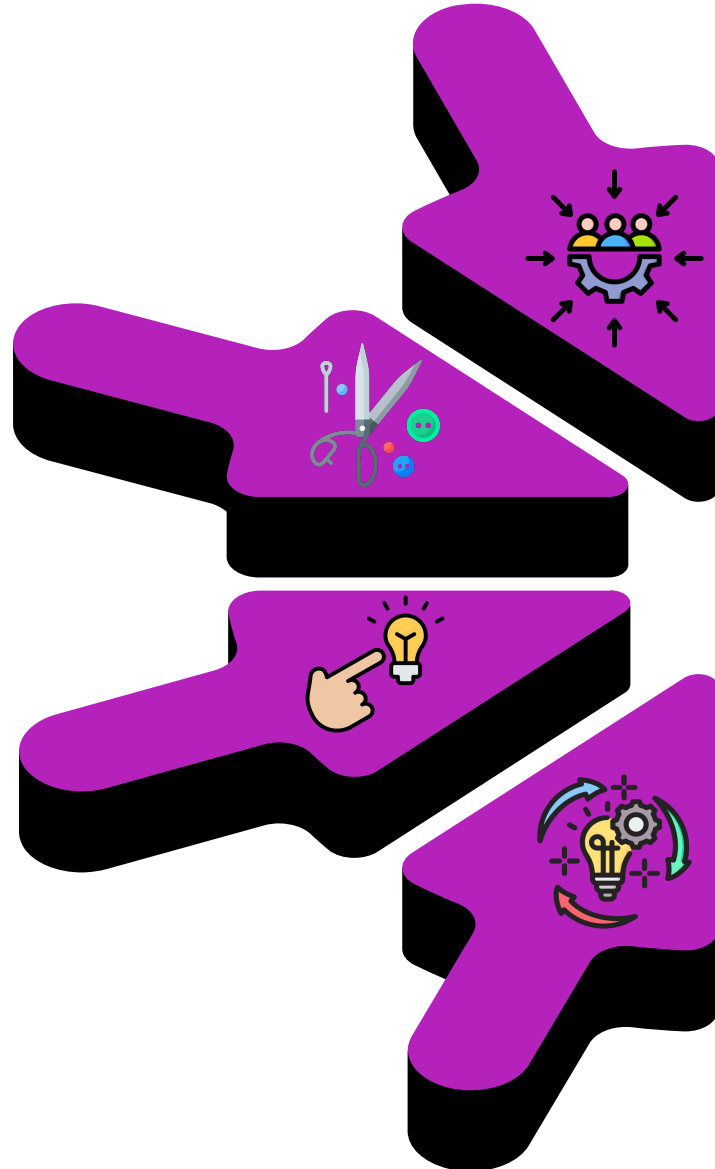
SCALABILITY

- Provide a scalable platform that grow with organization
- Ensuring service quality for new teams

PLATFORM AS A PRODUCT?



TREAT YOUR PLATFORM AS A PRODUCT



Customer Centric

A product for the developers

Tailor-made

Carefully designed and curated

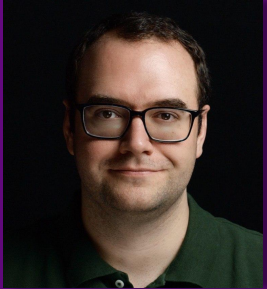
Simplicity

Simplify some workflow, abstract some details, and provide easier user interfaces

Continual Evolution

Take advantage of technology changes

THANK YOU!



Max Körbächer

Founder & Cloud Native Advisor



← Happy to connect on LinkedIn



Hossein Salahi

Principal Platform Engineer



← Happy to connect on LinkedIn

Icons: <https://www.flaticon.com/> created by smashingstocks