PLATFORM ENGINEERING UND DIE ZUKUNFT DER INTERNEN PLATTFORMPRODUKTE

Max Körbächer | Founder & Cloud Native Advisor

Core Principles and Promises of DevOps

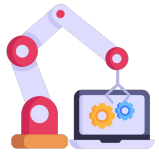
Collaboration

Break Down Silos

Foster a communication Culture

Cross-Functional Teams

Organizational-wide eng agement



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

REPLY LIQUID

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 t he problem

Building DevOps Team

Elimination of bottlenecks

Breaking down silos

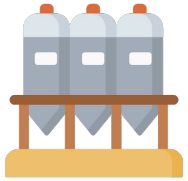
But, yet another silo!

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

REPLY LIQUID

WHY DEVOPS FAILS?

Why may development teams not adopt it?

- Dev Teams are to slow (not their fault)
- Not everyone dreams about end-to-end responsibility!
- Too many new tools, for too many problems to solve
- Dev Teams were not part of the service conceptualization
- Service discovery and ownership
- Container platforms are not the best abstraction layer

The economical damage

As reference a 60 people strong dev team:

- **average yearly costs of 4.5m**

- **ca. 10% of time is blocked by repeating work**
- **ca. 10% of time is wasted in resolving second party errors and bugs**
- **ca. 5% of time is trashed due to local/remote deployment issues**

- **15% of the team quits/joins demanding around 1 month for onboarding**

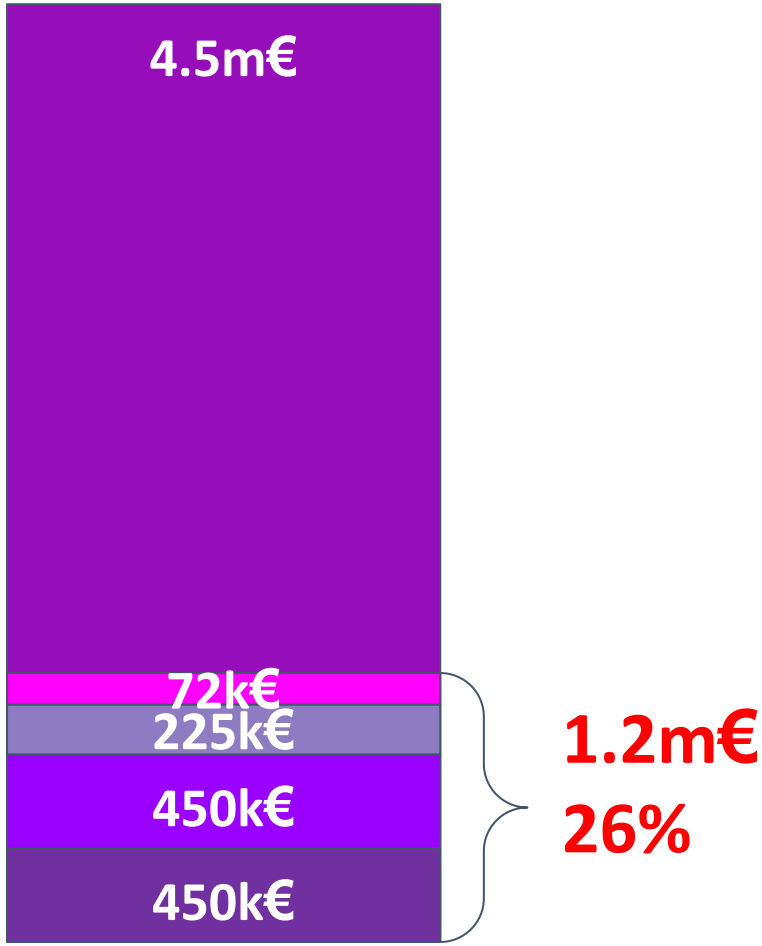
The economical damage

As reference a 60 people strong dev team:

- **average yearly costs of 4.5m**

- **ca. 10% of time is blocked by repeating work**
- **ca. 10% of time is wasted in resolving second party errors and bugs**
- **ca. 5% of time is trashed due to local/remote deployment issues**

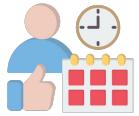
- **15% of the team quits/joins demanding around 1 month for onboarding**



REPLY LIQUID

PLATFORM ENGINEERING

Definition



A discipline of designing and building toolchains and workflows

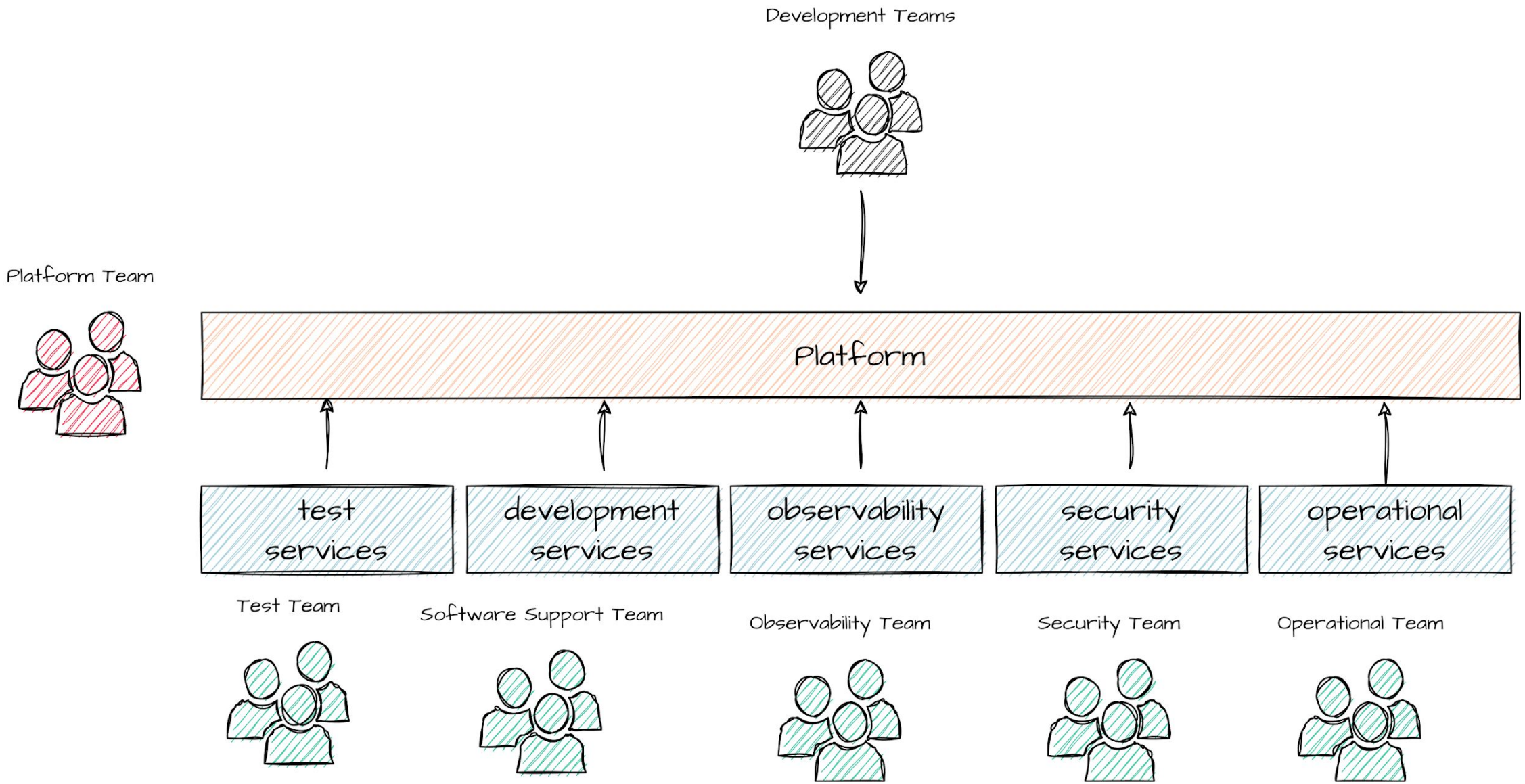
A layer abstraction to simplify underlying infrastructure provisioning and management

A standardized, automated, scalable environments

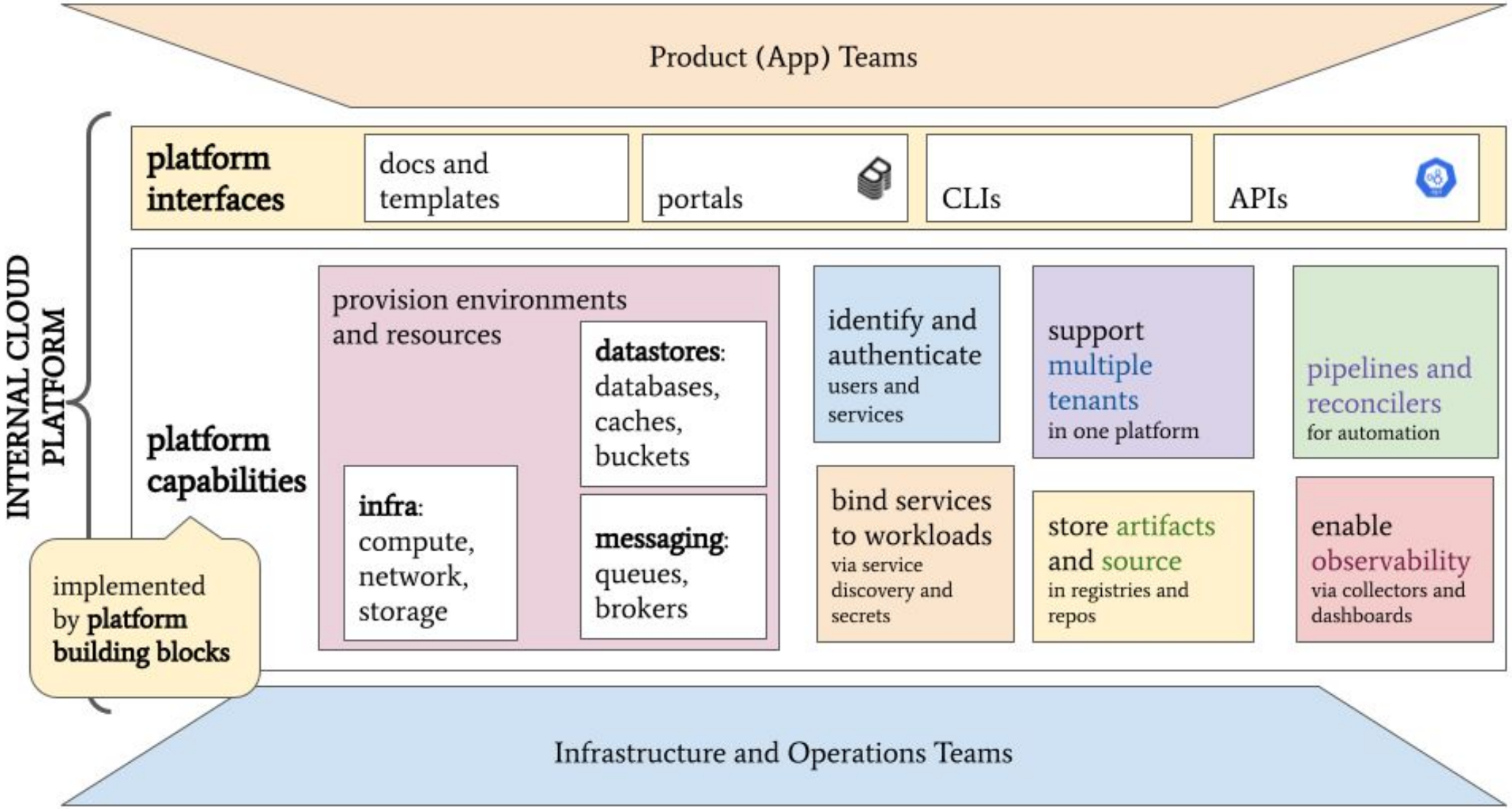
Self-service capabilities for development teams

More efficient application delivery

REPLY LIQUID



https://www.innoq.com/en/articles/2023/08/is-platform-engineering-the-new-devops/#discoverabilityandownership



https://blog.joshgav.com/posts/kubecon-platforms-review

PLATFORM ENGINEERING RESPONSIBILITIES

Infrastructure Automation	Workflow Orchestration
Standardization	Self-Service Capabilities
Security and Compliance	Observability
Scalability and Resilience	Collaboration and Feedback

CRAFT YOUR OWN PLATFORMS

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

REPLY
LIQUID

PLATFORM LAYERS

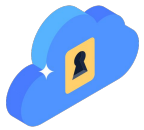
Consist of multiple components that interact in different speed with each other

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

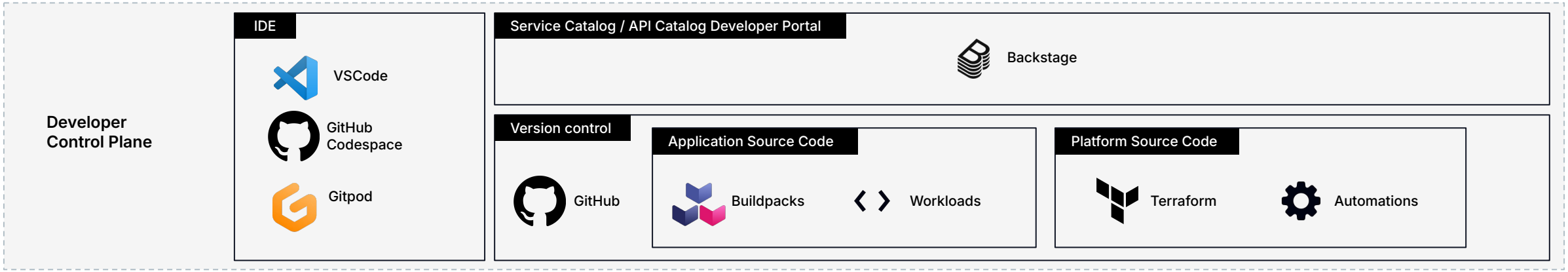
Usually a cloud provider, IaaS or hypervisor

Capability Plane

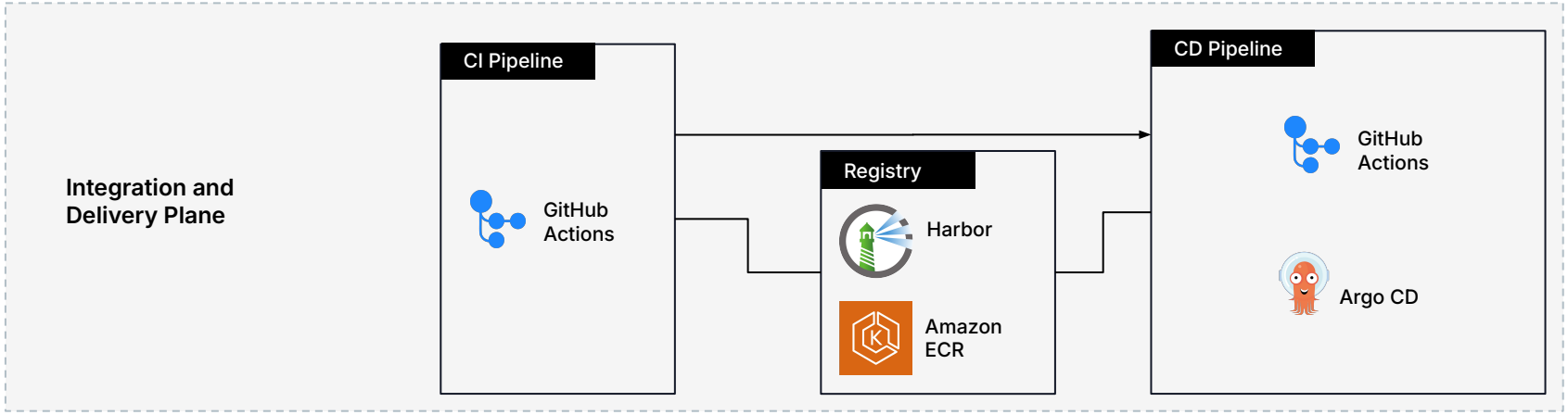
Features which the platform provides for the users in lives “within” the platform

REPLY LIQUID

IDP REFERENCE



IDP REFERENCE



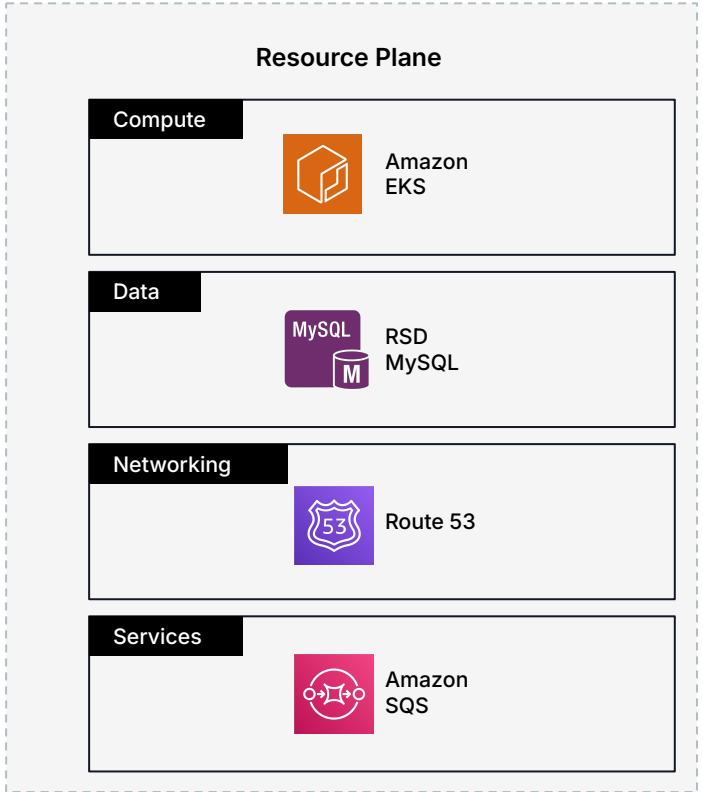
IDP REFERENCE

REPLY LIQUID

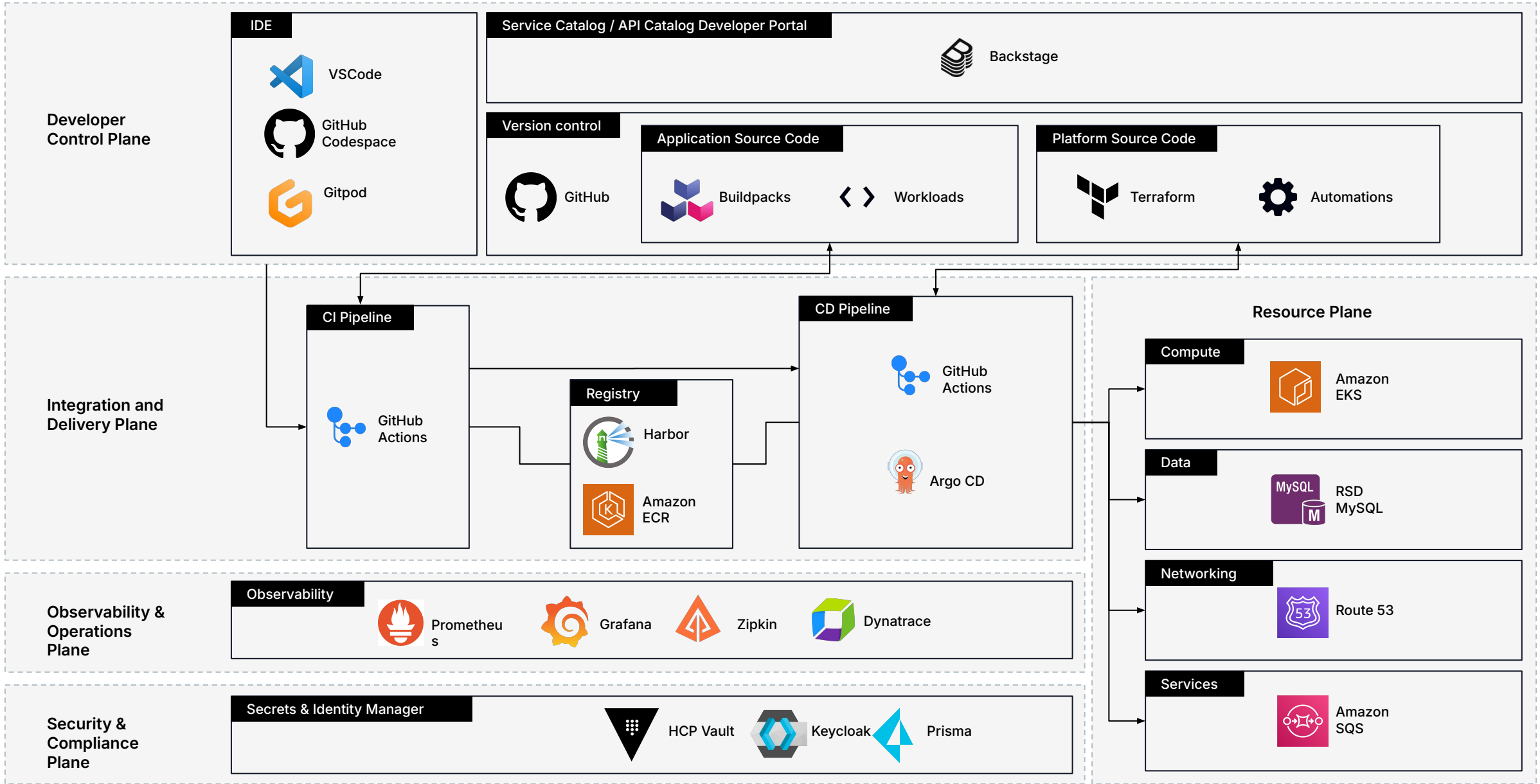
IDP REFERENCE

REPLY
LIQUID

IDP REFERENCE

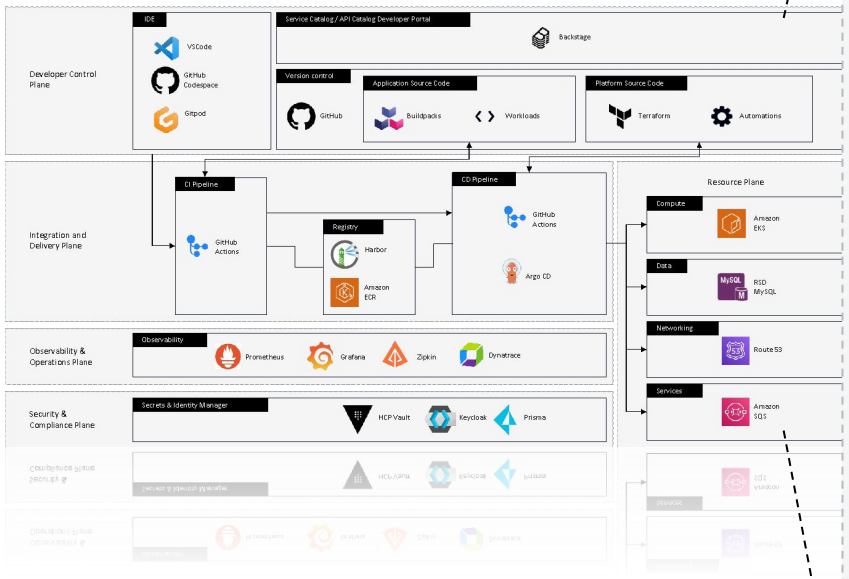


IDP REFERENCE



Source: platformengineering.org

IDP REFERENCE



Capability Plane

User Space

dapr

Scale & Schedule

Kn

KEDA

kube-green

Network

External DNS for kubernetes

NGINX

cilium

Heavy Babies

kafka

Kubeflow

VOLCANO
Kubernetes Native Batch System

Resource Integration

Crossplane

CONTAINER STORAGE INTERFACE

GPU Device Plugin

Security & Compliance

HashiCorp Vault

Kyverno

kube-hunter

Falco

Observability

OpenTelemetry

fluentbit

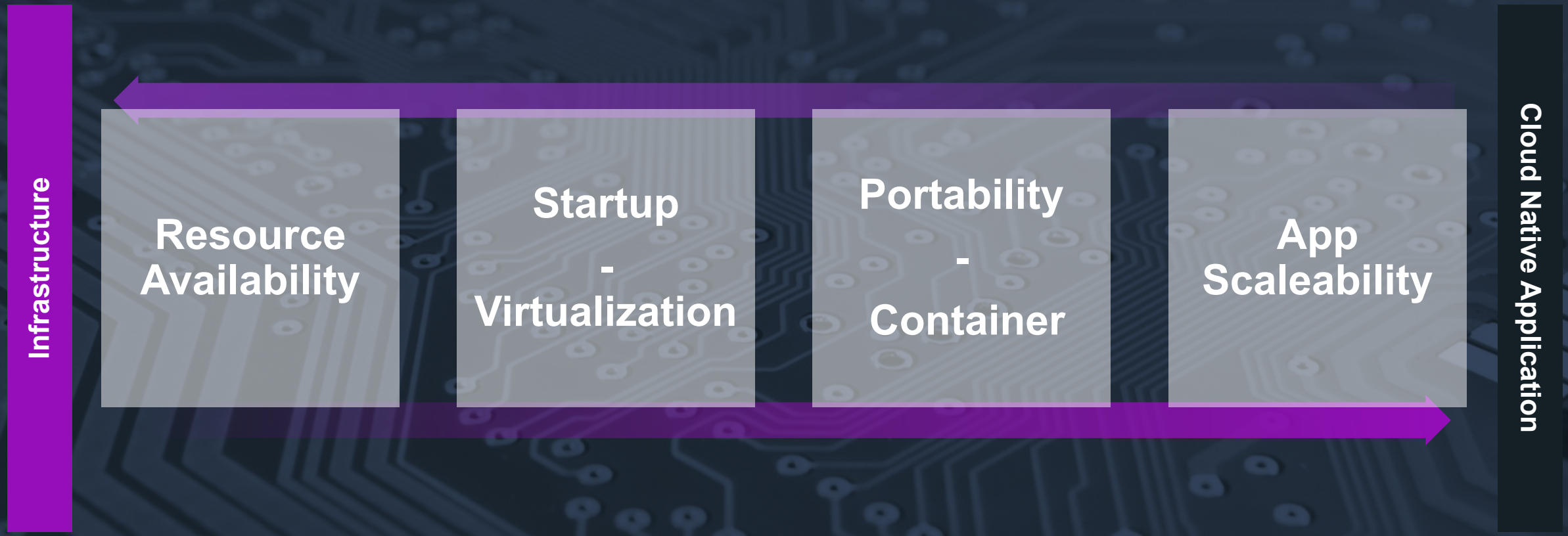
kubecost

Prometheus

REPLY LIQUID

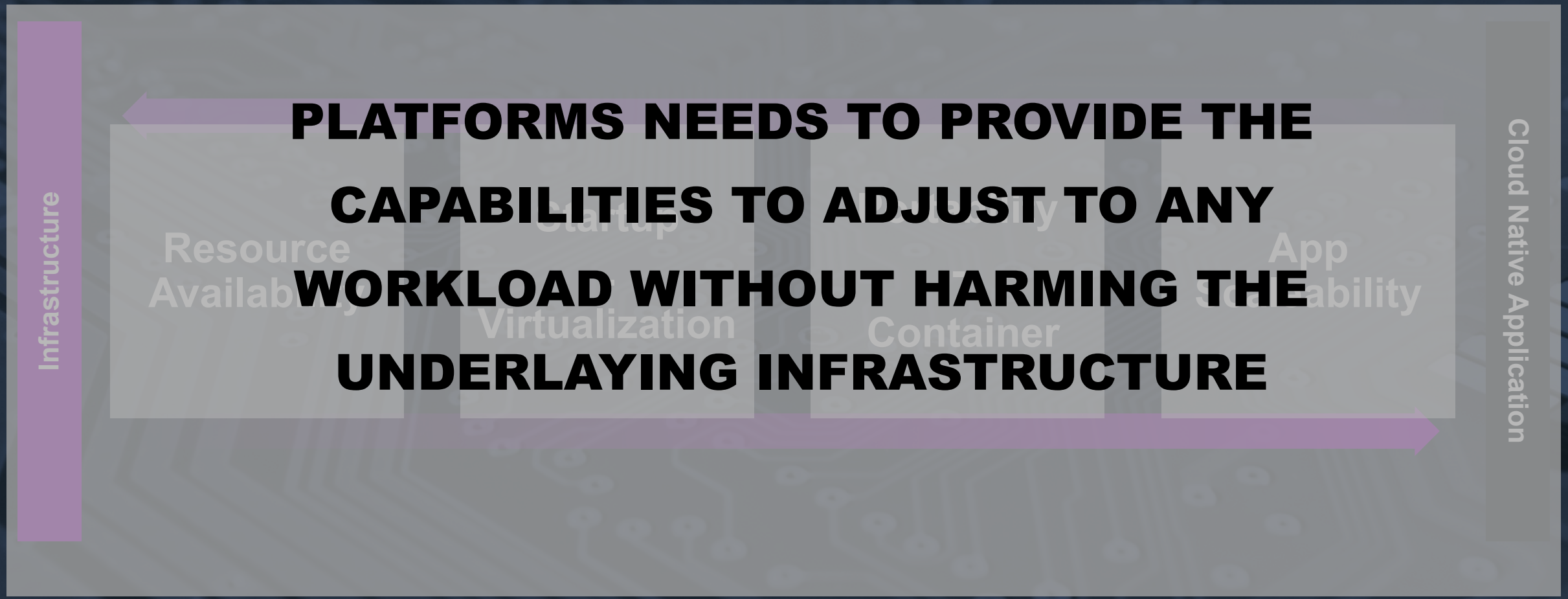
INFLUENCING

Hardware and software effects each other



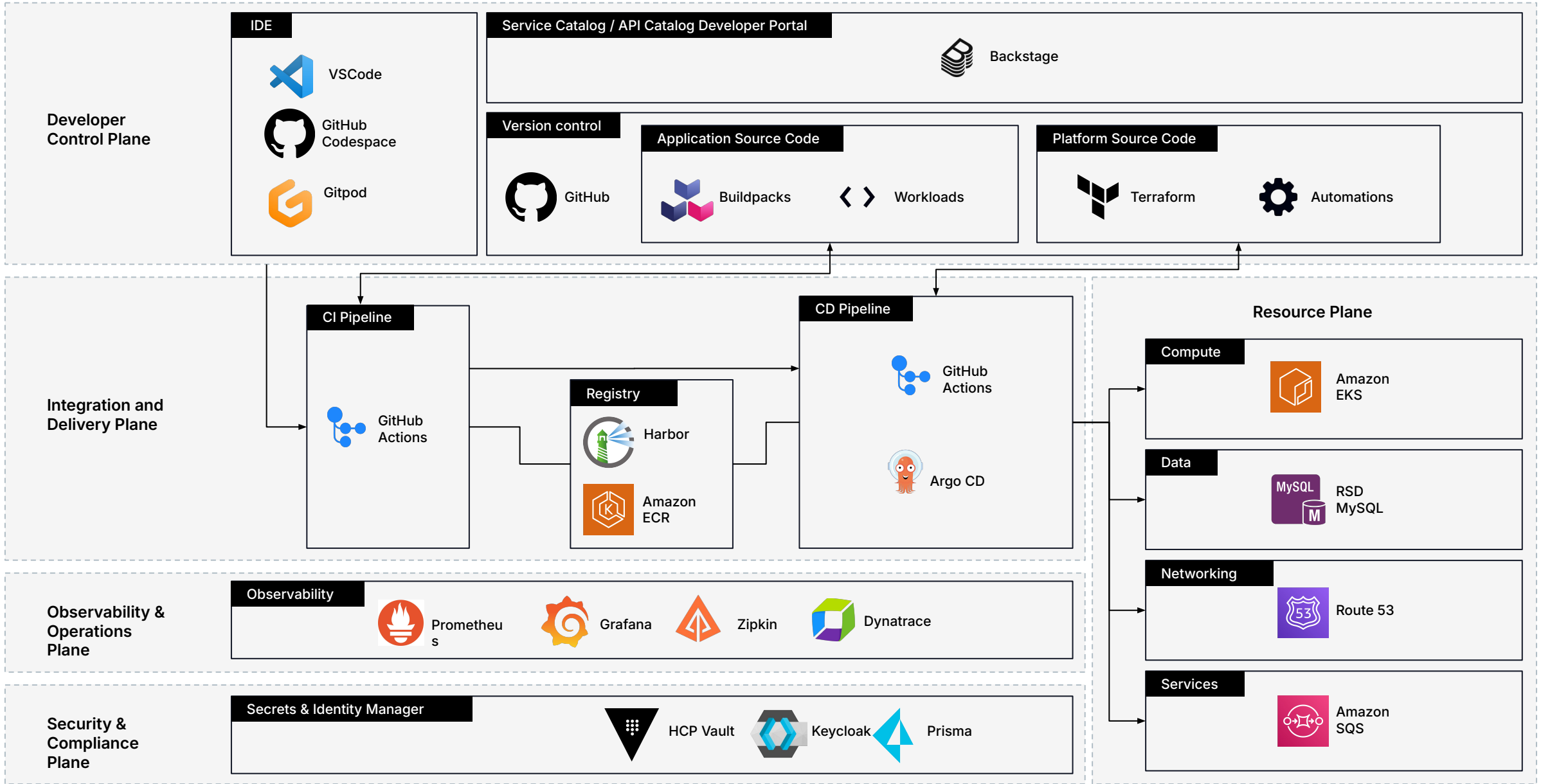
INFLUENCING

Hardware and software effects each other

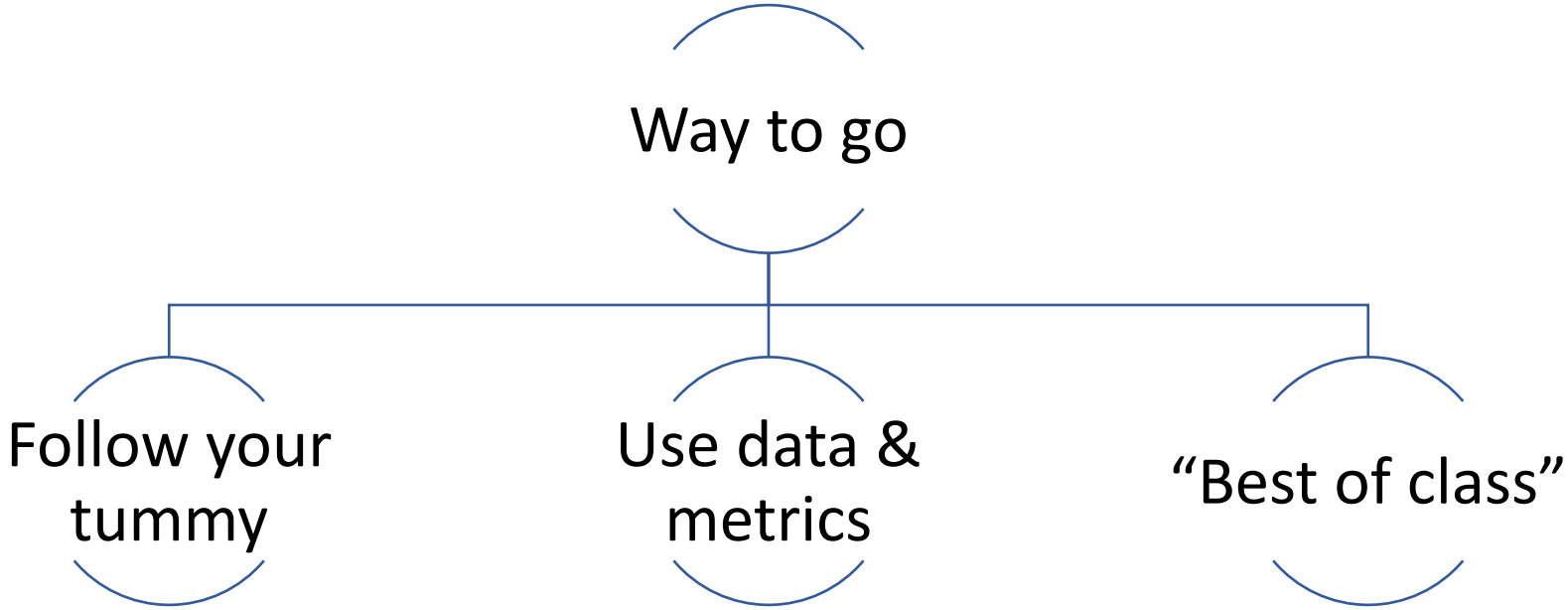


PLATFORMS NEEDS TO PROVIDE THE CAPABILITIES TO ADJUST TO ANY WORKLOAD WITHOUT HARMING THE UNDERLAYING INFRASTRUCTURE

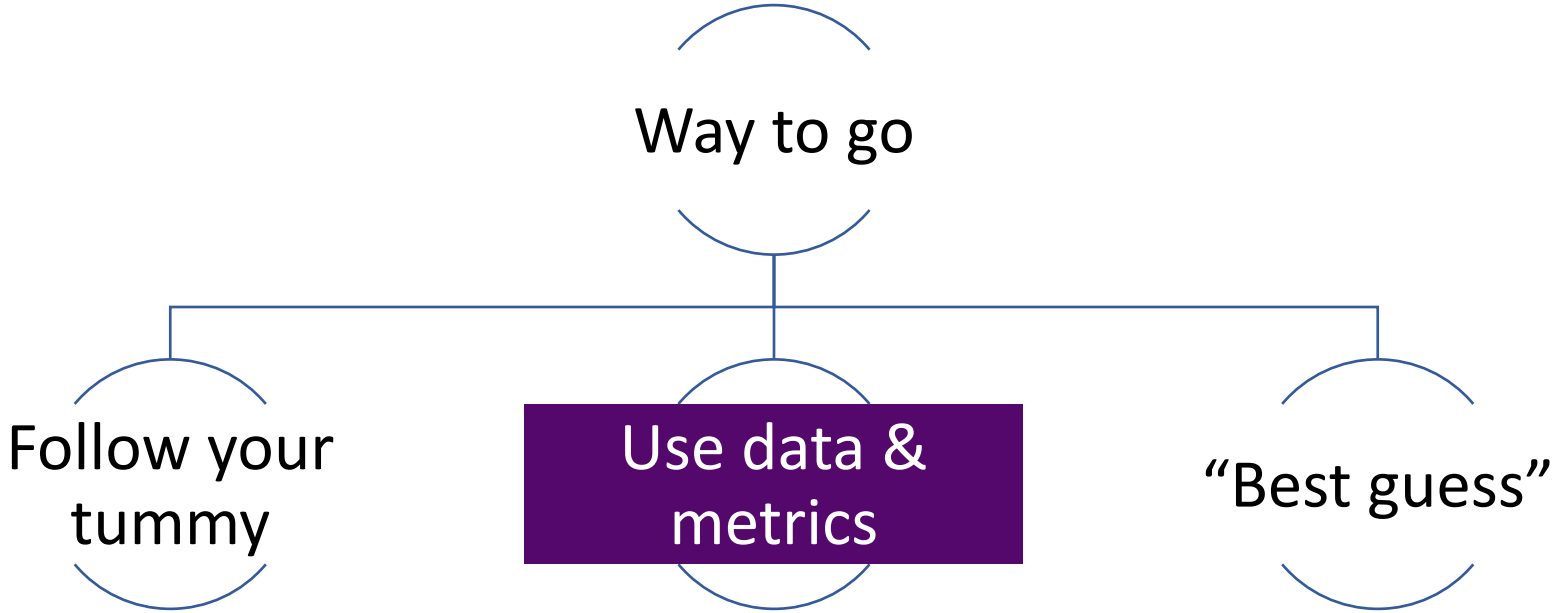
IDP REFERENCE



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 %

REPLY LIQUID

DEFINE GOLDEN PATH

Collect data on certain observation points

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

Source: platformengineering.org

REPLY LIQUID

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%

Source: platformengineering.org

DATA EXAMPLE TO FIND GOLDEN PATH

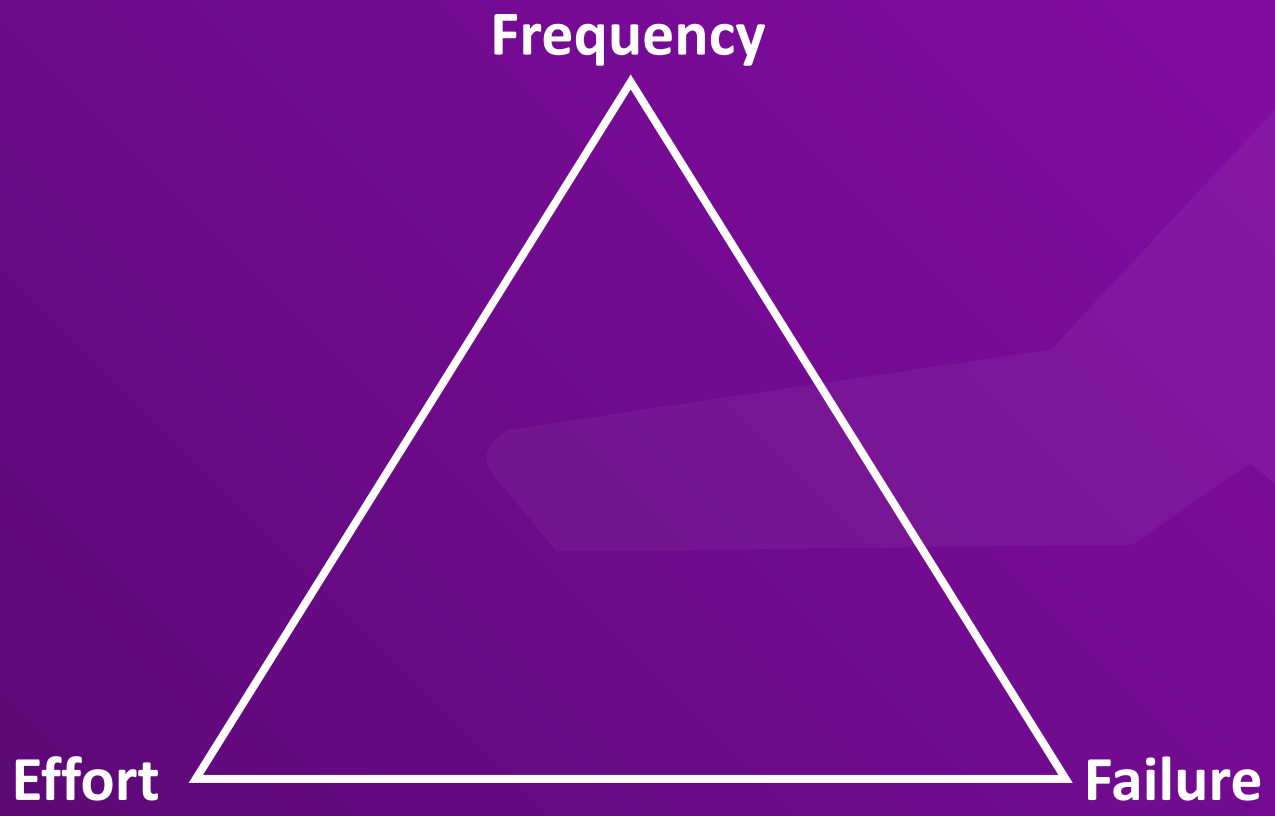
Dev Steps	Frequency %	Dev Time h	Ops Effort h	Alignment/ Comms h	Failure Rate %
Add/Update Services	40%	16	8	2	30%
Add/Update Resources	20%	8	24	3	6%
Add/Update Configurations	60%	1	1	0	10%
Change architecture	3%	60	20	10	60%
Create new environments	4%	24	24	3	25%
Onboarding new developers	20%	80	16	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

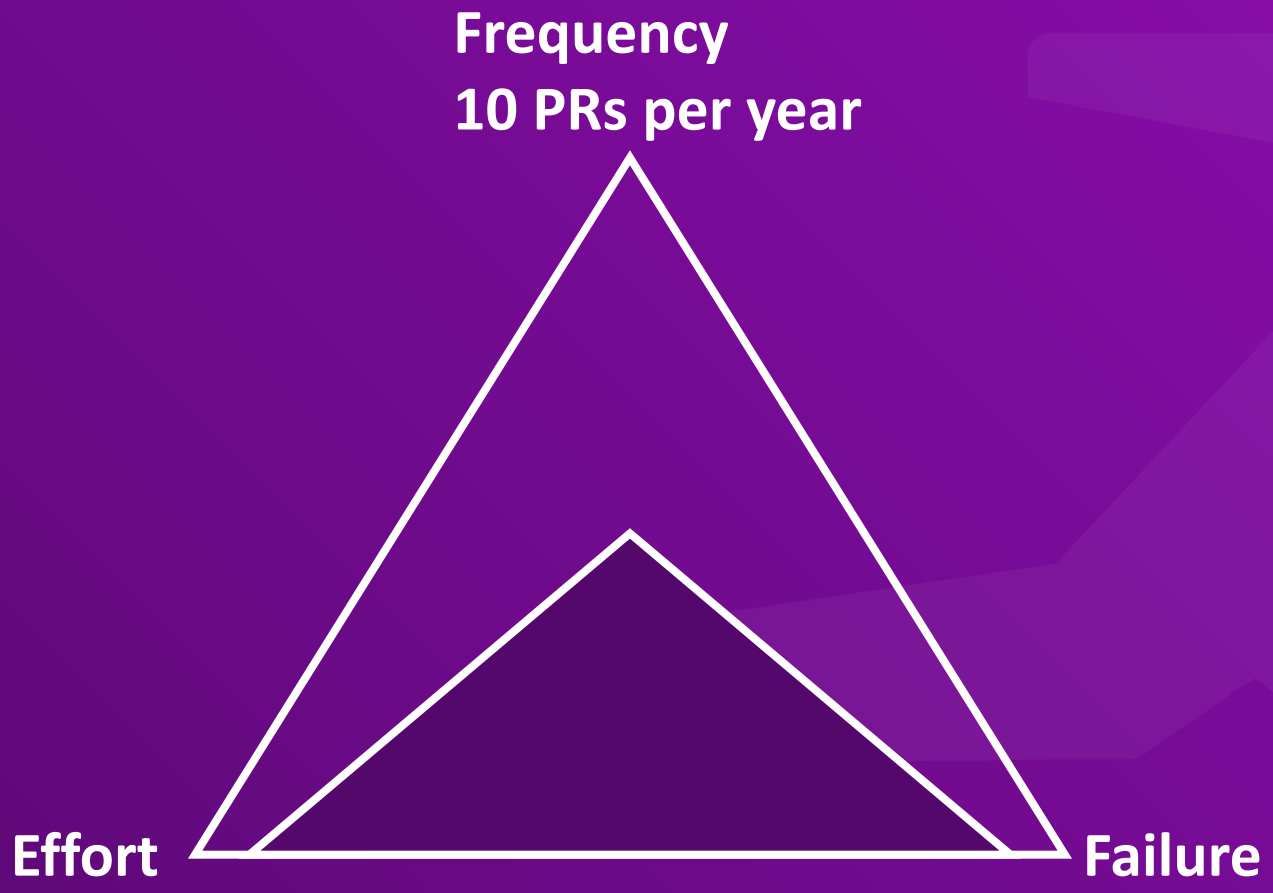
Best 3 per KPI

REPLY LIQUID

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%

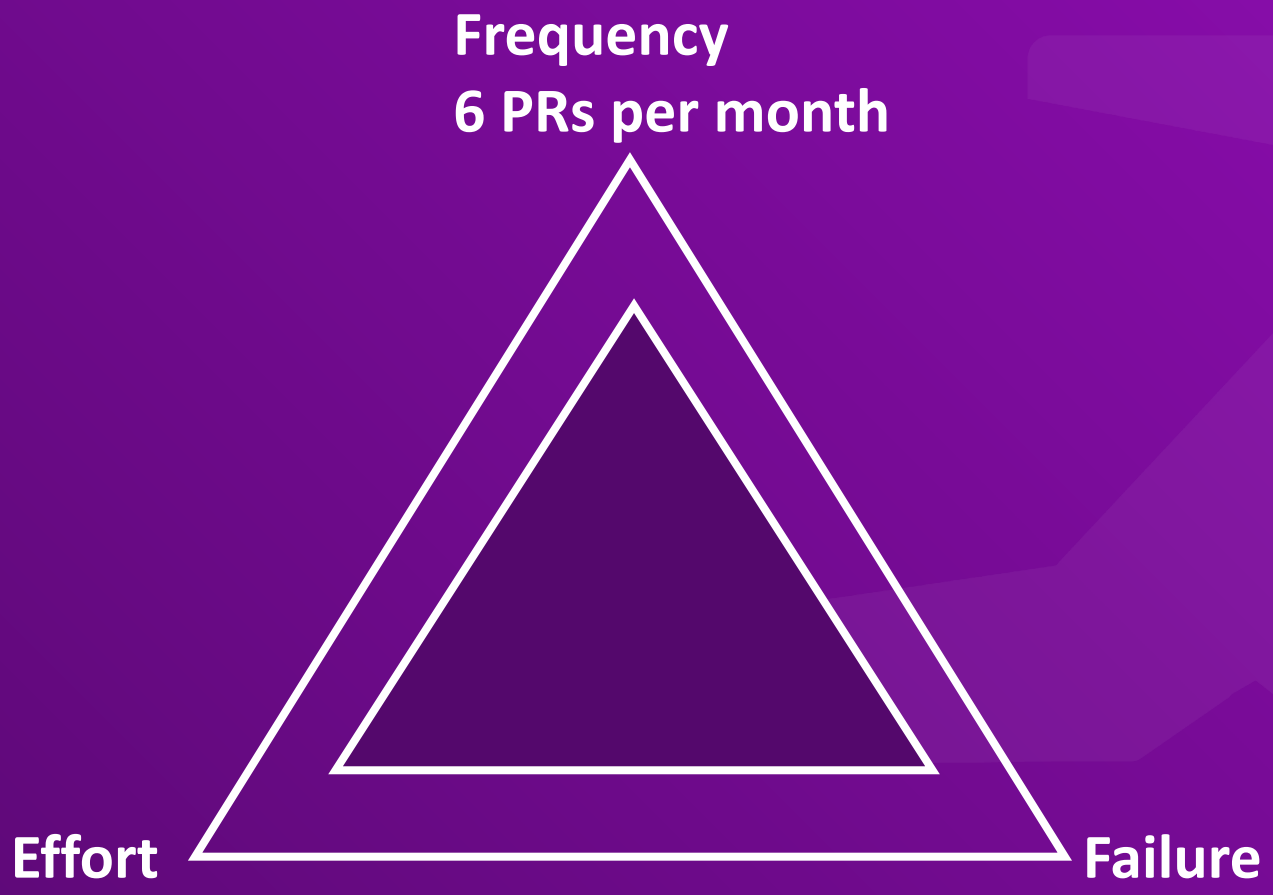
Worst 3 per KPI

Best 3 per KPI

Golden Path

REPLY LIQUID

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%

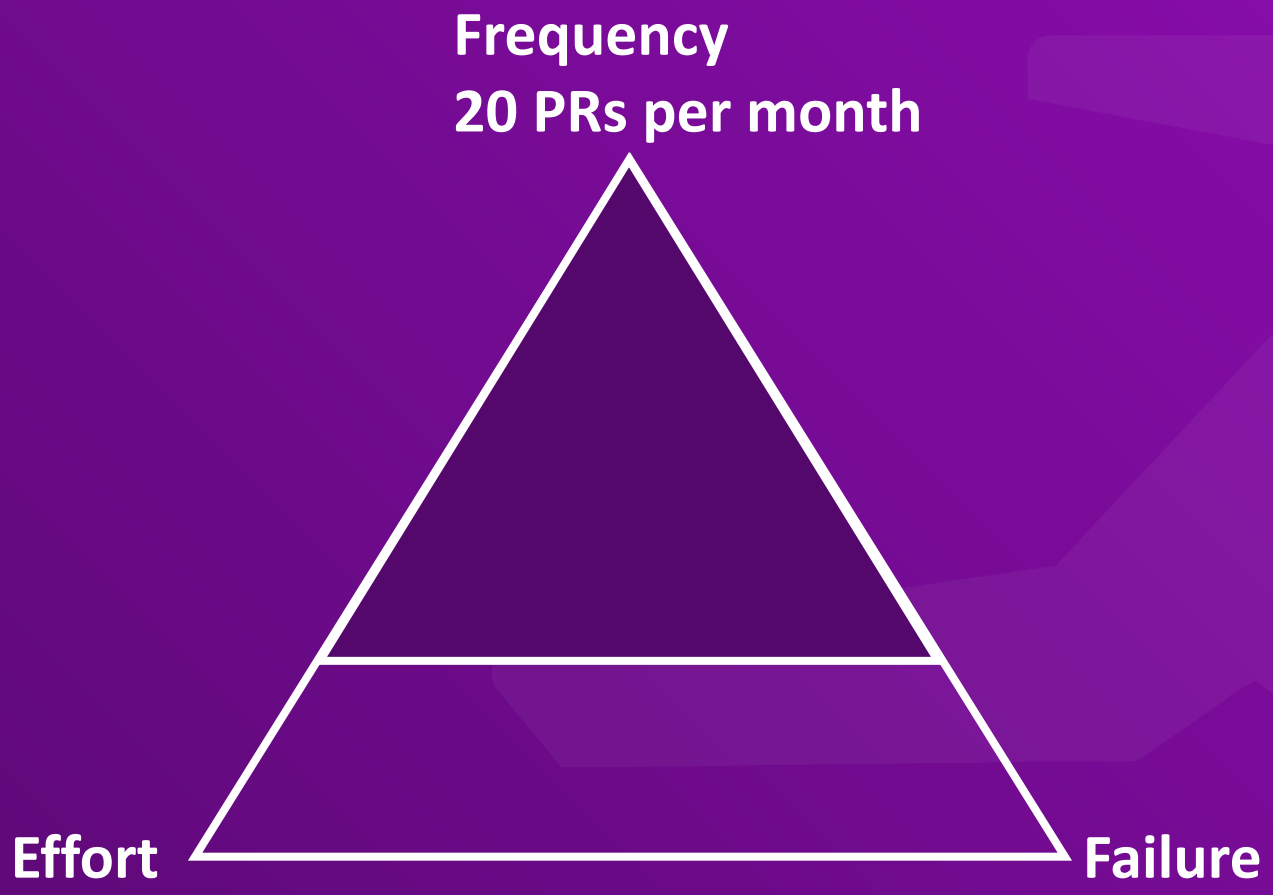
Worst 3 per KPI

Best 3 per KPI

Golden Path

REPLY LIQUID

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

REPLY LIQUID

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.

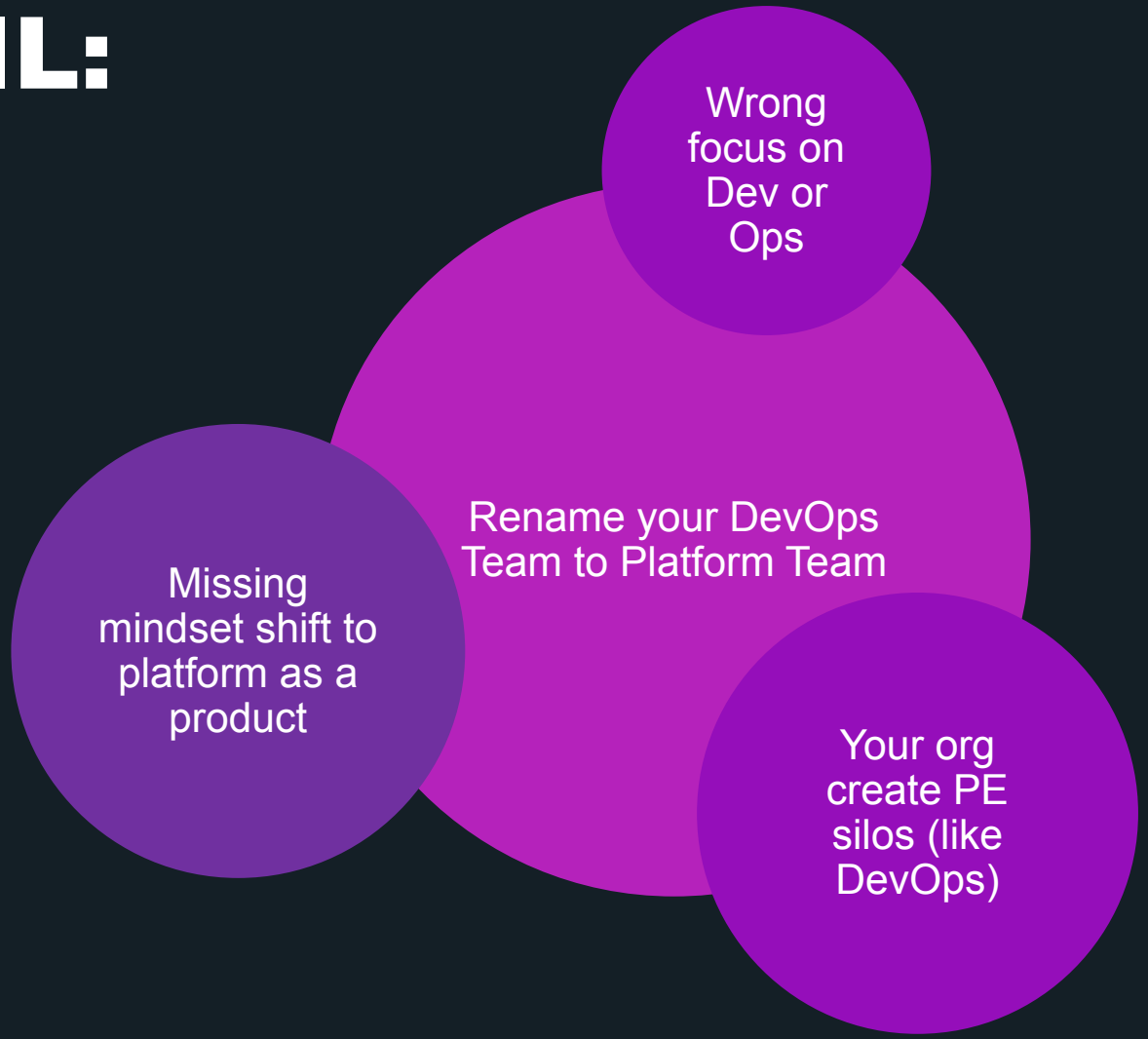
But, responsibilities can be shared.

REPLY LIQUID

ELSE, YOU WILL FAIL:

Technical “Playground”

- 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?

STANDARDIZATION

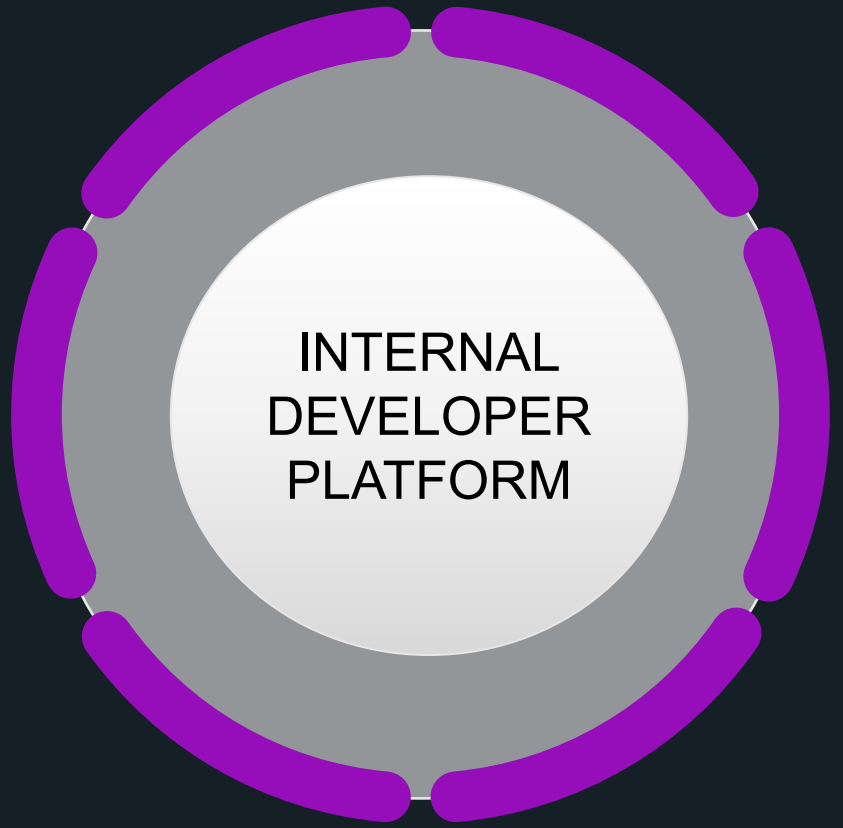
- A standard set of tools and services
- Reduces risk of inconsistencies and errors

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 compllance



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

Use Cases of Successful IDP

Spotify

- create 2x more code changes in 17% less time
- deploy 2x often their software
- single components are 3x less likely to be fixed
- contributed 2.3x more on GitHub
- Developers are 5% more likely to stay

Expedia

- Ship on first day of job with 95% success rate
- manage over 7.000 components without losing track
- Reduced context switching due to a single IDP

TOYOTA

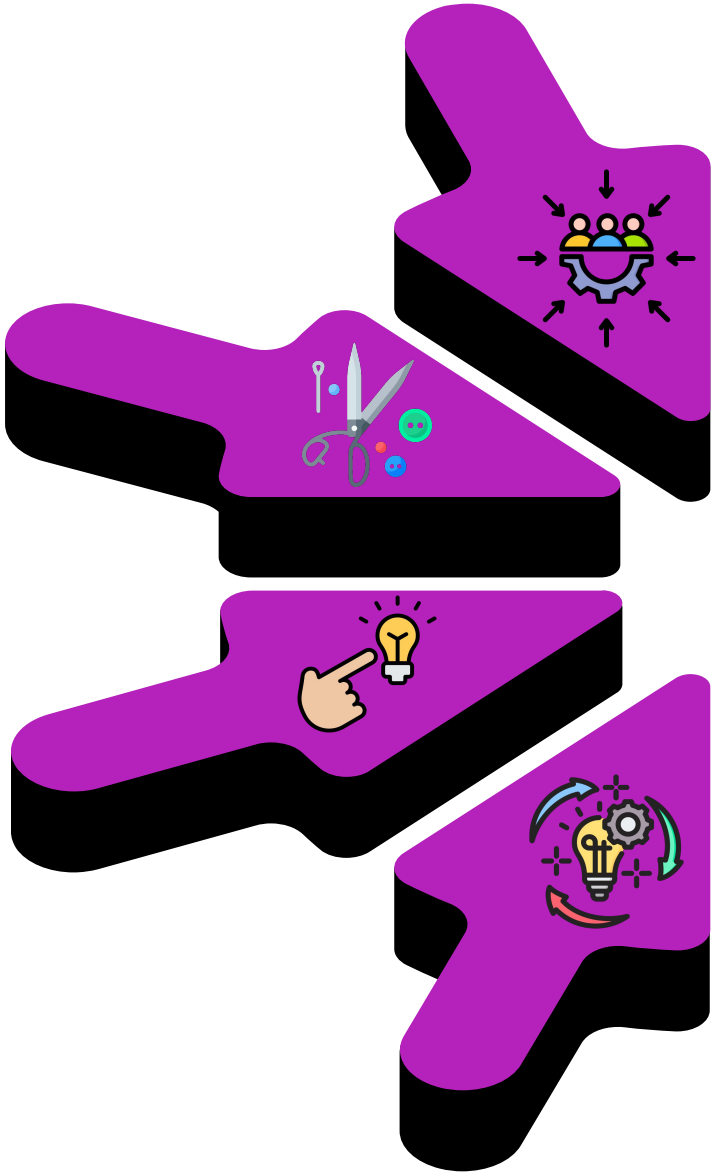
- shipping projects weekly instead of quarterly
- 8-12 weeks saved in time reducing costs of $5m year over year
- pre-approved templates speed up delivery

Source: Platformes Community

REPLY LIQUID

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

REPLY LIQUID

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