SQUER

Platform Engineering:
Unveiling Success for Autonomous Teams

David Leitner

SQUER

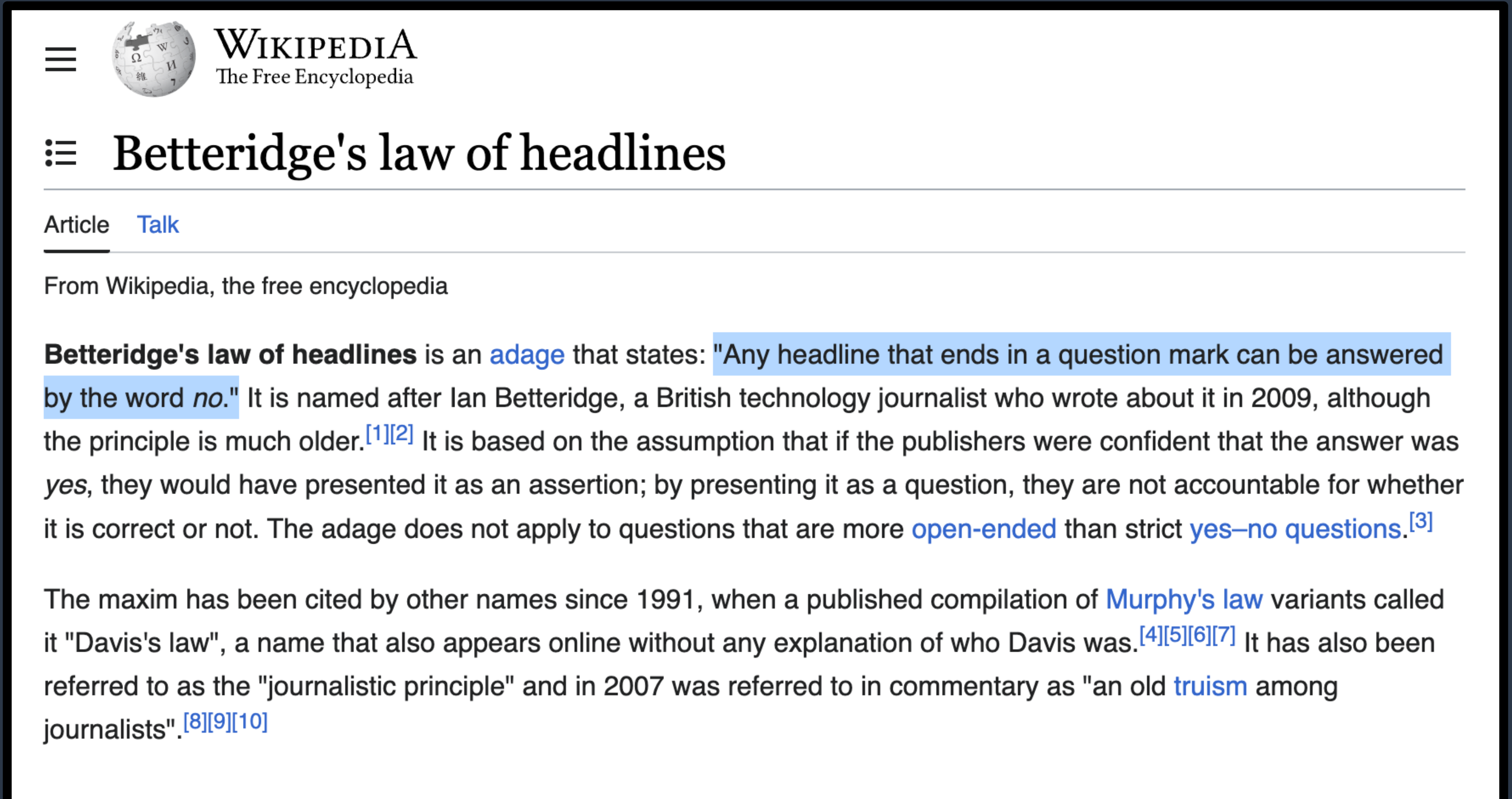
Platform Engineering:

Unveiling Success for Autonomous Teams?

David Leitner

SQUER

wikipedia.org/wiki/Betteridge%27s_law_of_headlines



WIKIPEDIA
The Free Encyclopedia

Betteridge's law of headlines

Article Talk

From Wikipedia, the free encyclopedia

Betteridge's law of headlines is an adage that states: "Any headline that ends in a question mark can be answered by the word *no*." It is named after Ian Betteridge, a British technology journalist who wrote about it in 2009, although the principle is much older.[1][2] It is based on the assumption that if the publishers were confident that the answer was *yes*, they would have presented it as an assertion; by presenting it as a question, they are not accountable for whether it is correct or not. The adage does not apply to questions that are more open-ended than strict yes–no questions.[3]

The maxim has been cited by other names since 1991, when a published compilation of Murphy's law variants called it "Davis's law", a name that also appears online without any explanation of who Davis was.[4][5][6][7] It has also been referred to as the "journalistic principle" and in 2007 was referred to in commentary as "an old truism among journalists".[8][9][10]

@duffleit

SQUER

DAVID LEITNER

Principal Engineer

SQUER

👋 david@squer.io

🌎 @duffleit

Prompts / LLMs

Frameworks

Compilers

Virtual Machines

Operating Systems

Assembler

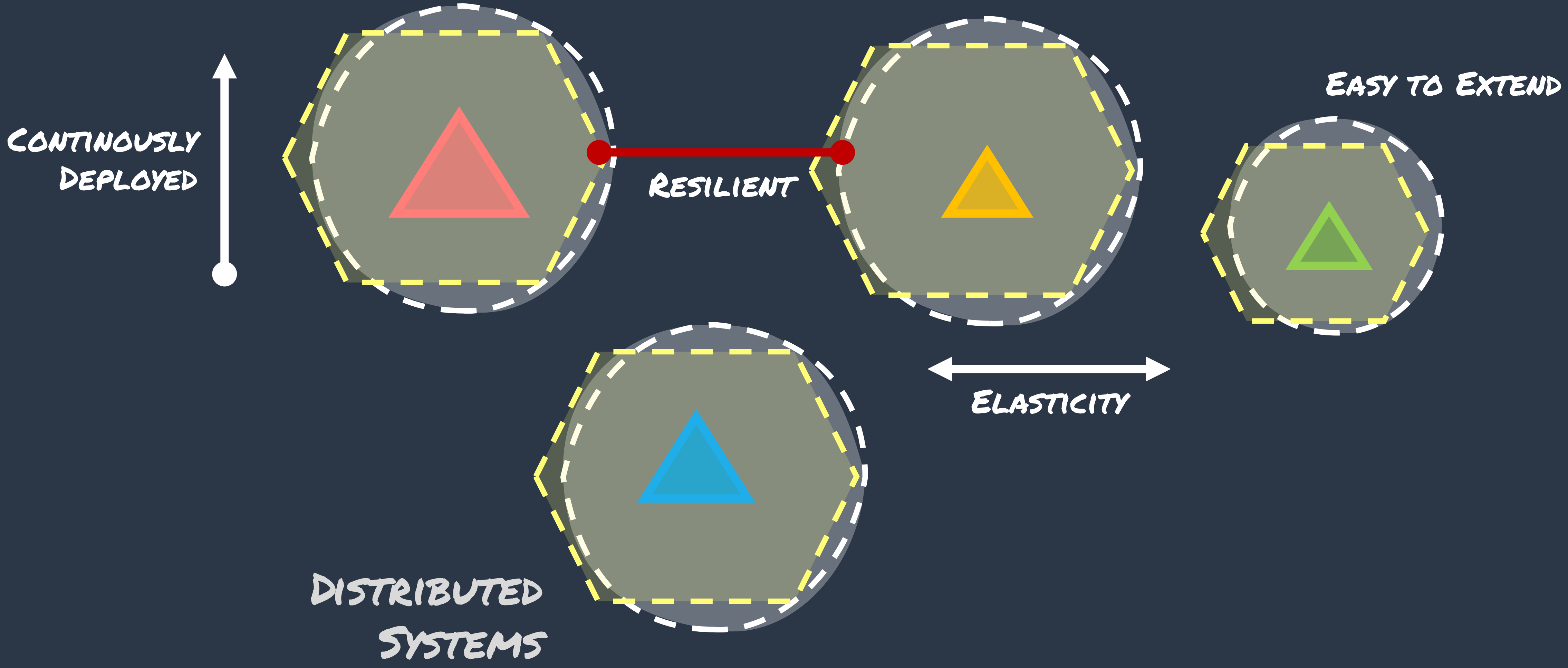
0 1 0 1 1 0 1 1

There is no other industry out there that is as good at introducing new abstractions as we are.

— *Ted Neward*

Monolithical Systems

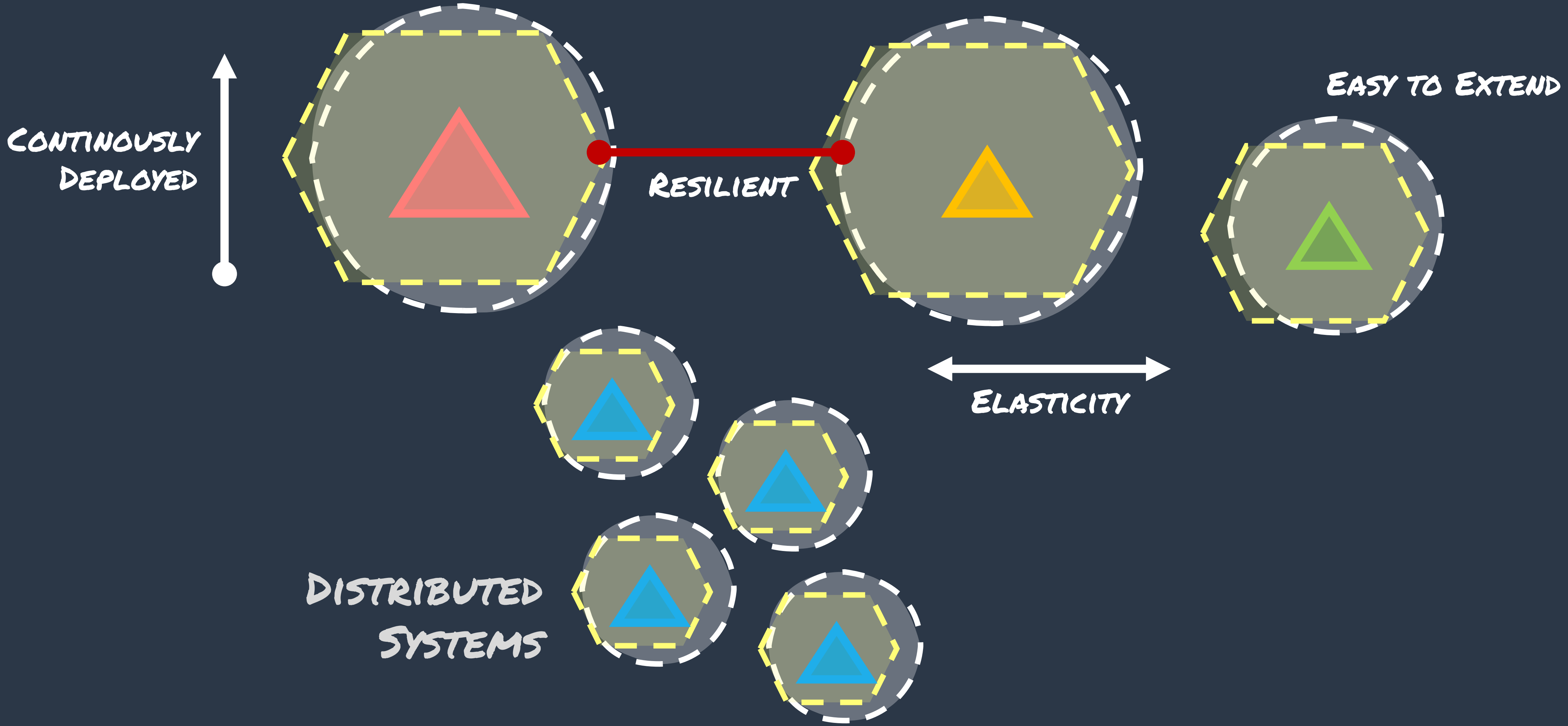
@duffleit



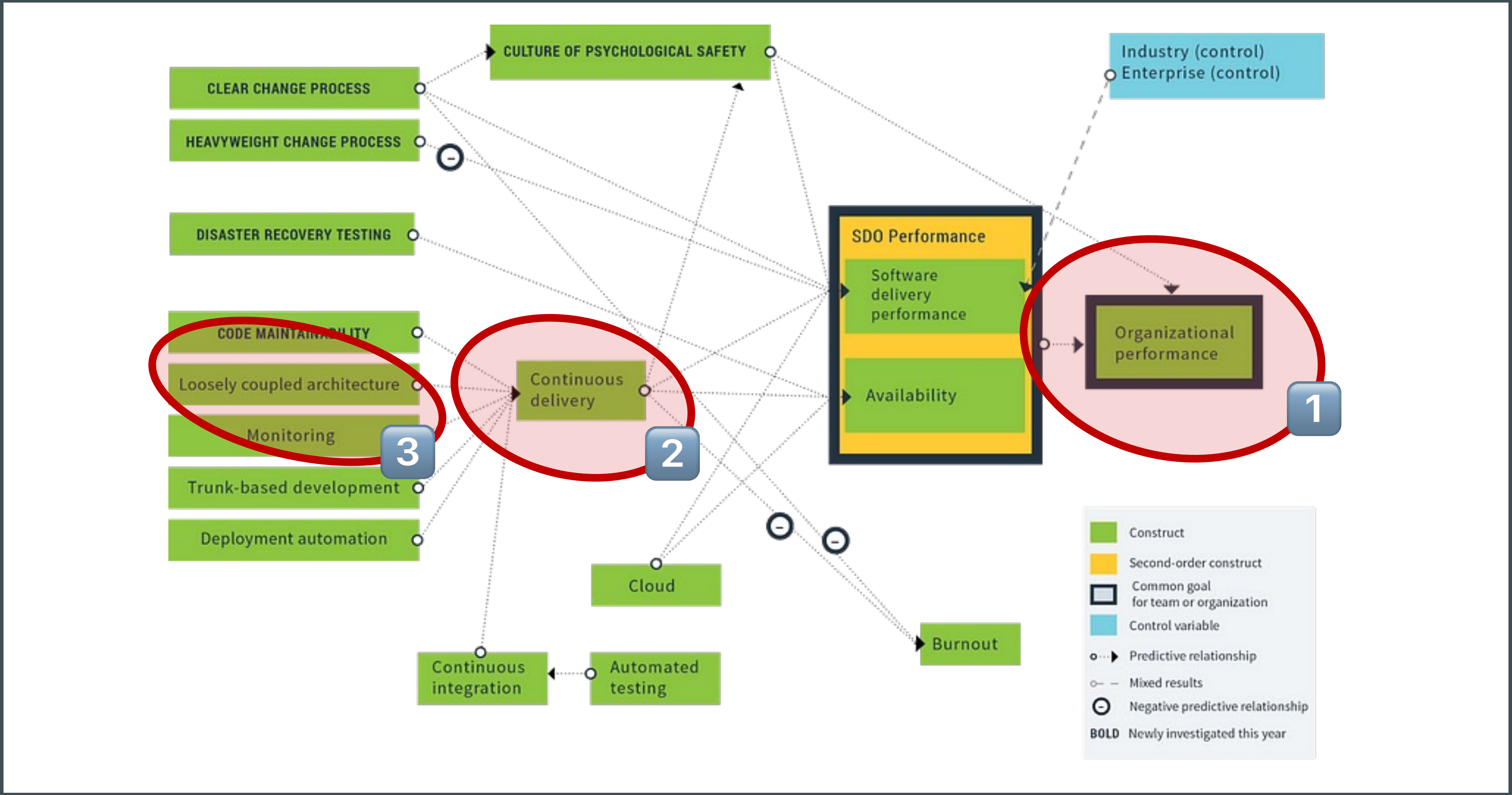
SQUER

@duffleit

SQUER



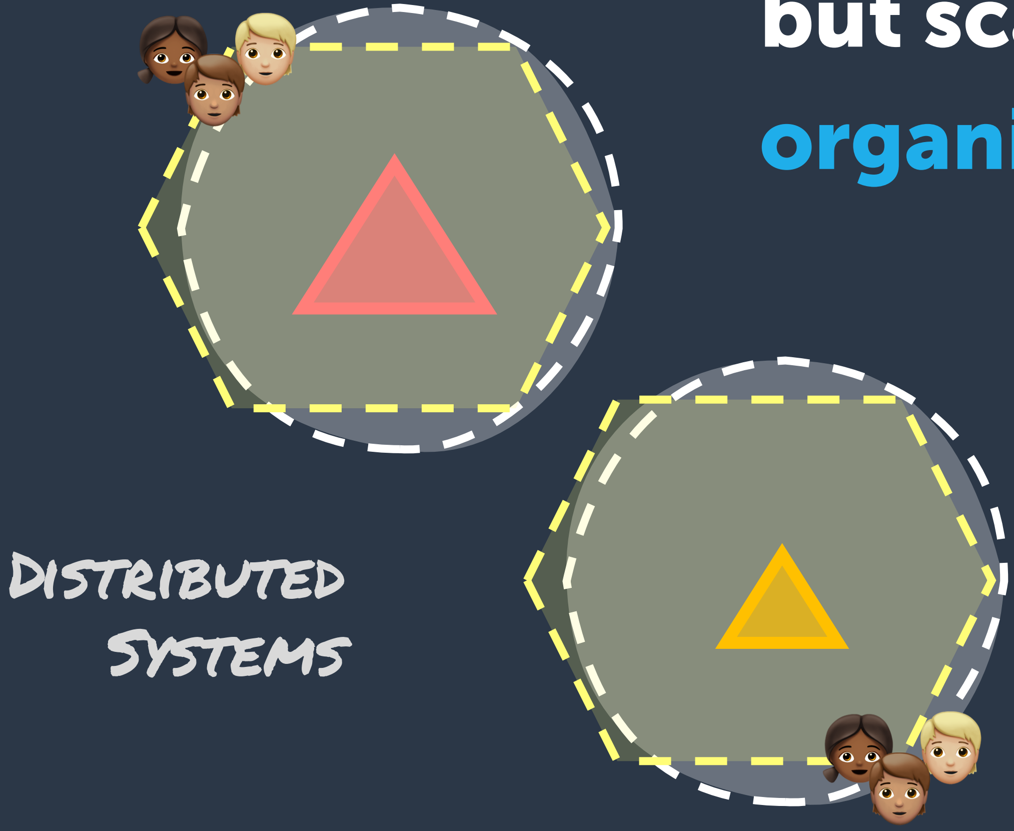
@duffleit



speakerdeck.com/squer/cultivating-architecture-in-an-agile-world-2019

SQUER

Microservices **aren't just about scaling your software architecture but scaling your teams. They are an organizational scaling strategy.**

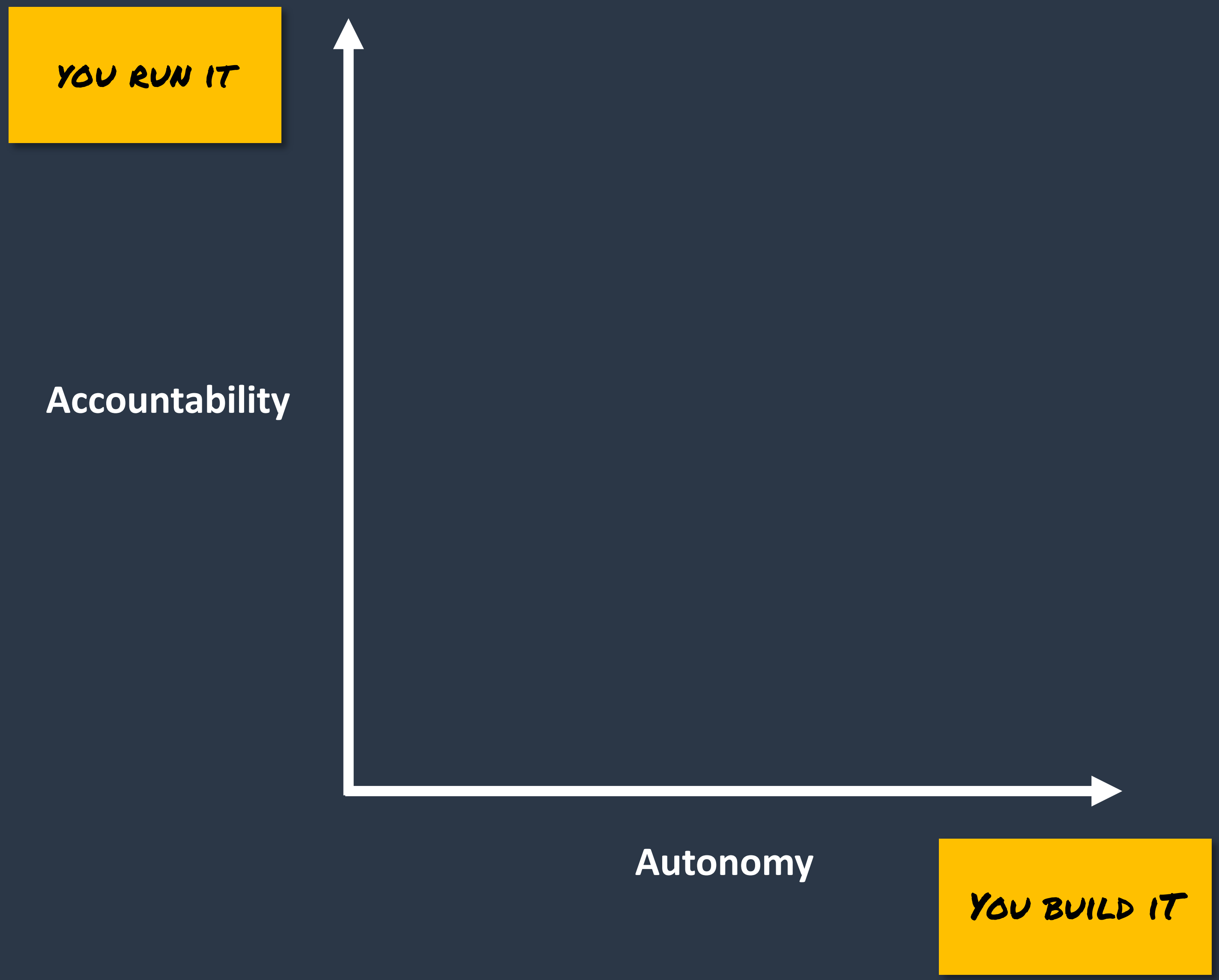


You build it, you run it.

— *Werner Vogels*

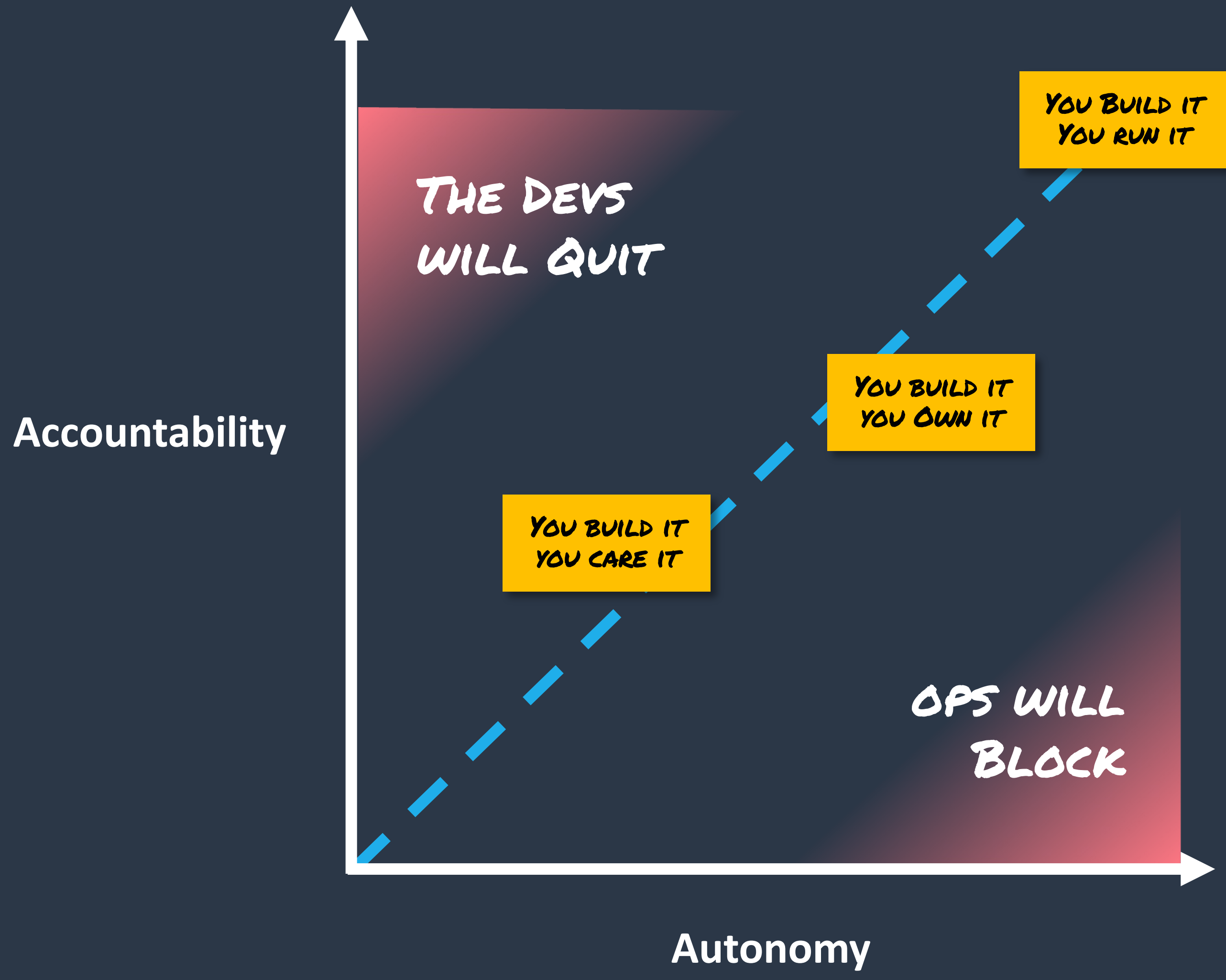
@duffleit

SQUER



@duffleit

SQUER

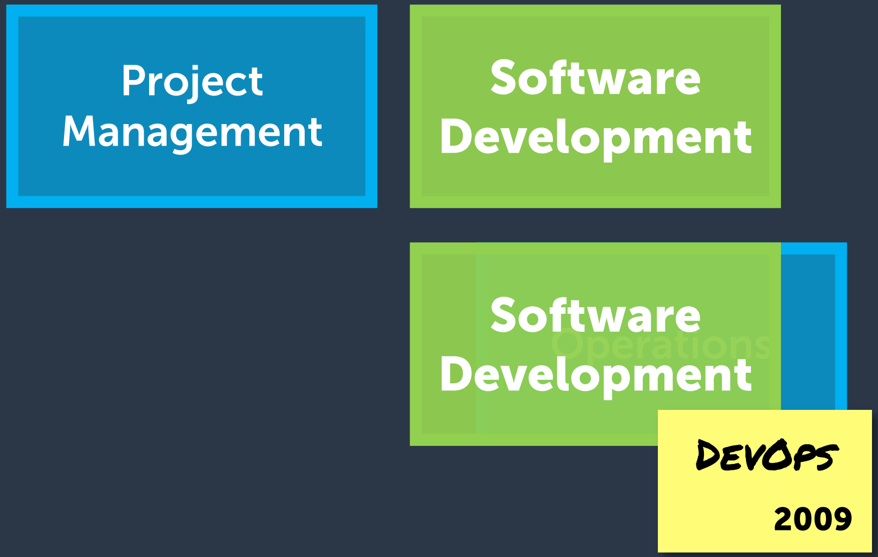


@duffleit

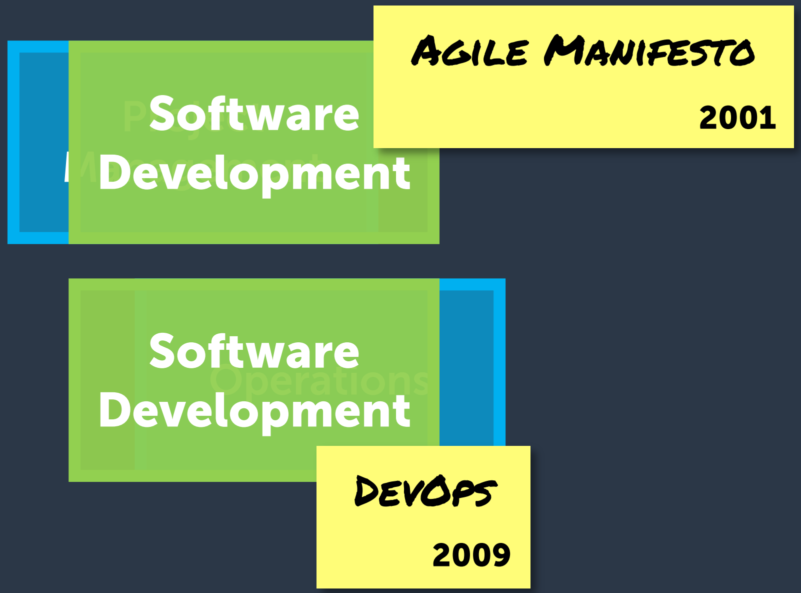
Software Development

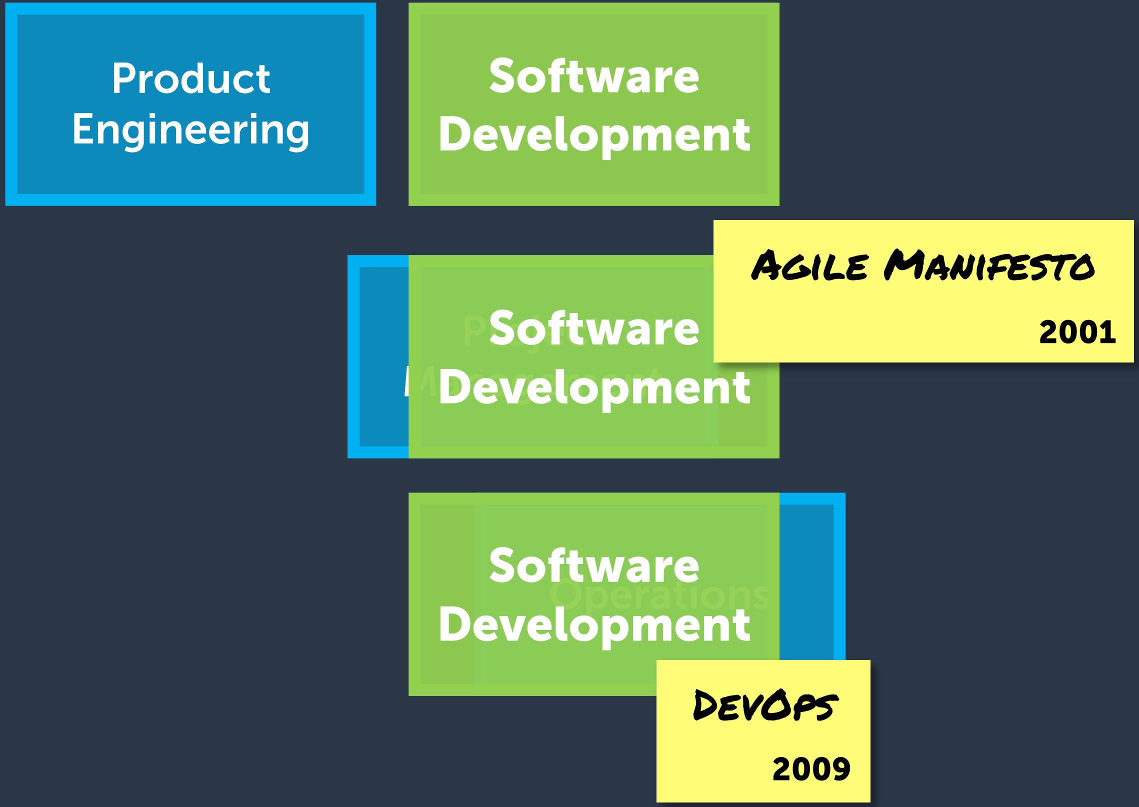
Operations

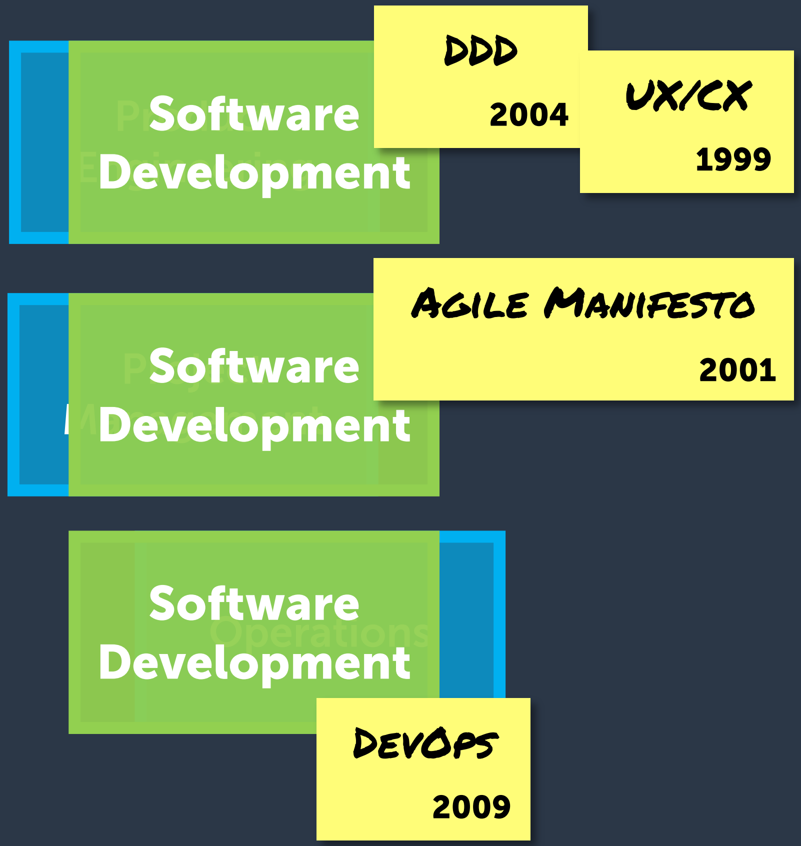
Software Development
Operations
DevOps
2009

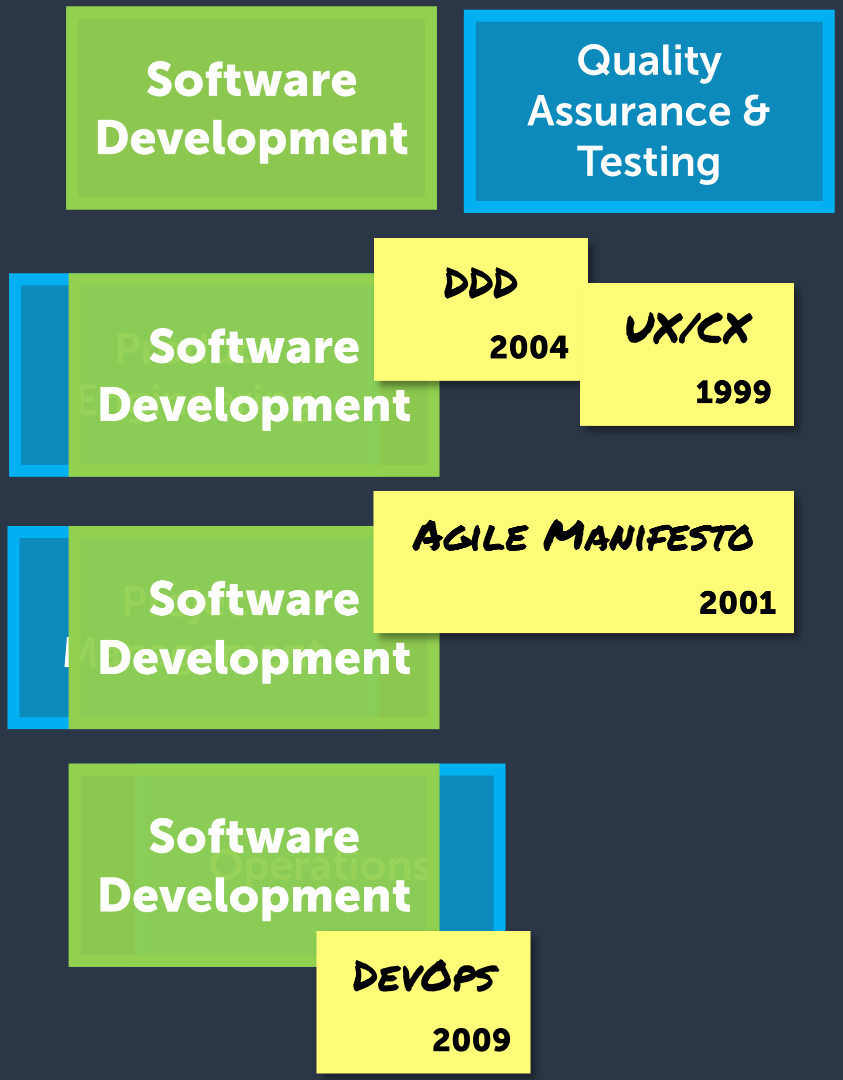

Project Management
Software Development
Software Development
Operations
DevOps
2009

SQUER

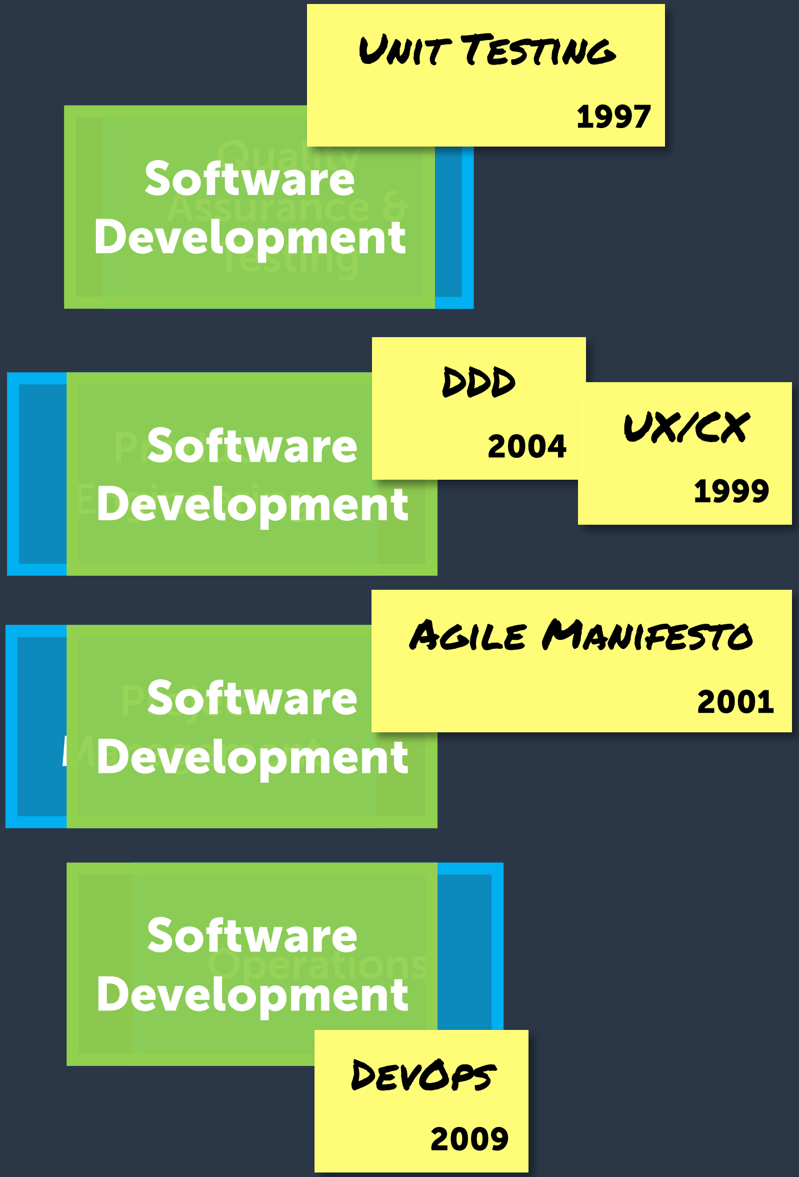

Software Development
Agile Manifesto
2001
Software Development
DevOps
2009


Product Engineering
Software Development
Software Development
Agile Manifesto
2001
Software Development
Operations
DevOps
2009


Software Development
DDD
2004
UX/CX
1999
Software Development
Agile Manifesto
2001
Software Development
DevOps
2009


Software Development
Quality Assurance & Testing
Software Development
DDD
2004
UX/CX
1999
Software Development
Agile Manifesto
2001
Software Development
DevOps
2009

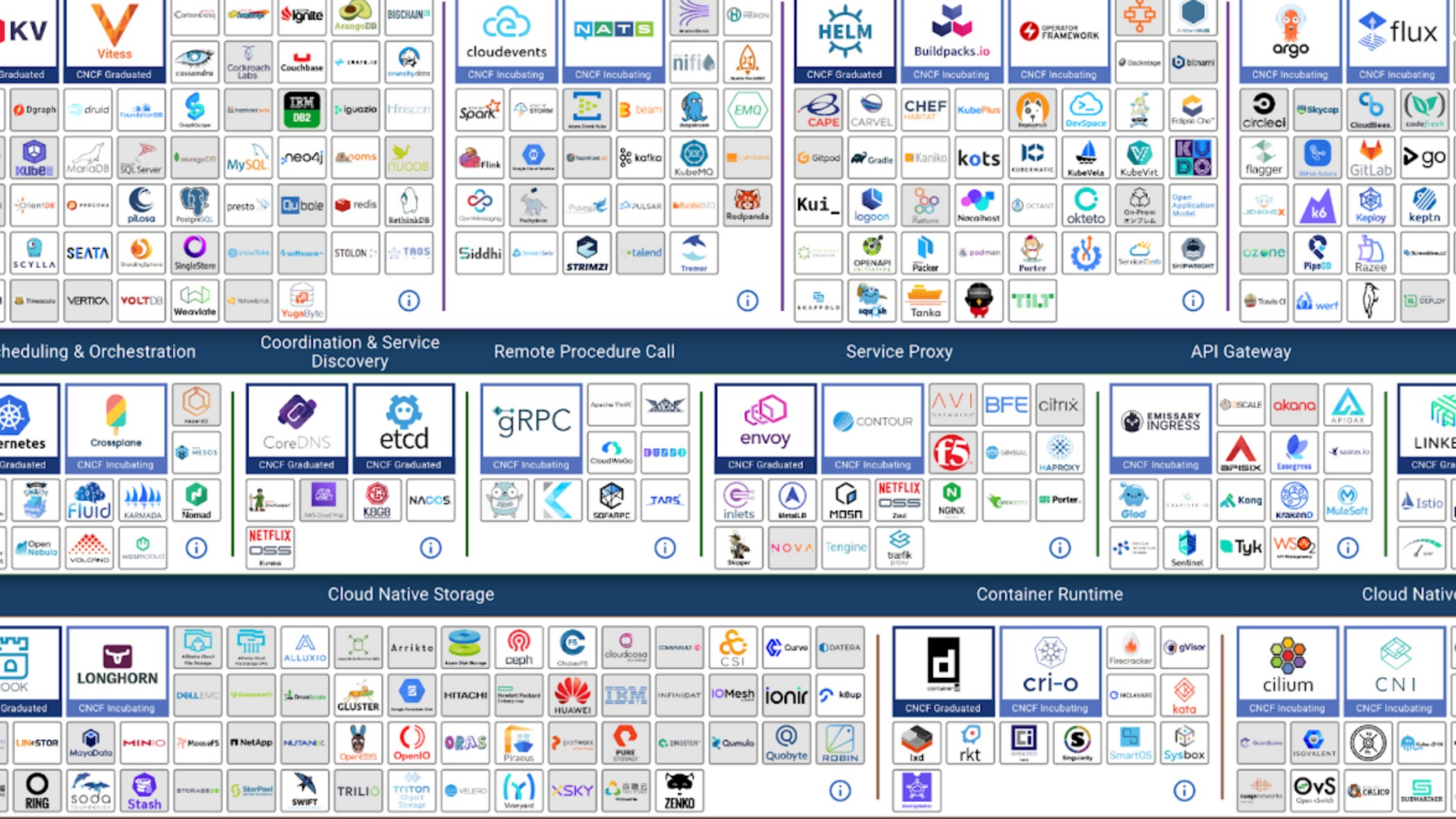
SQUER

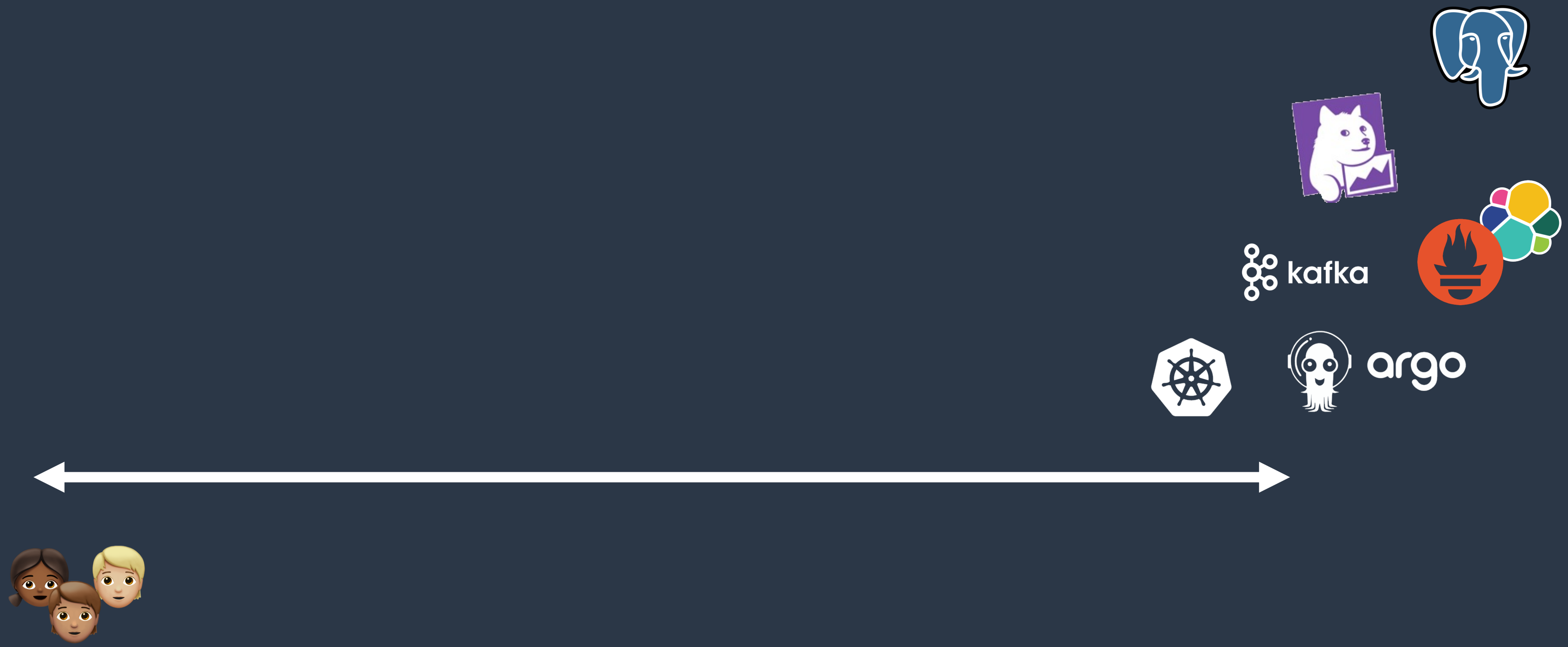


There is a constant strive for shift-left in our industry.

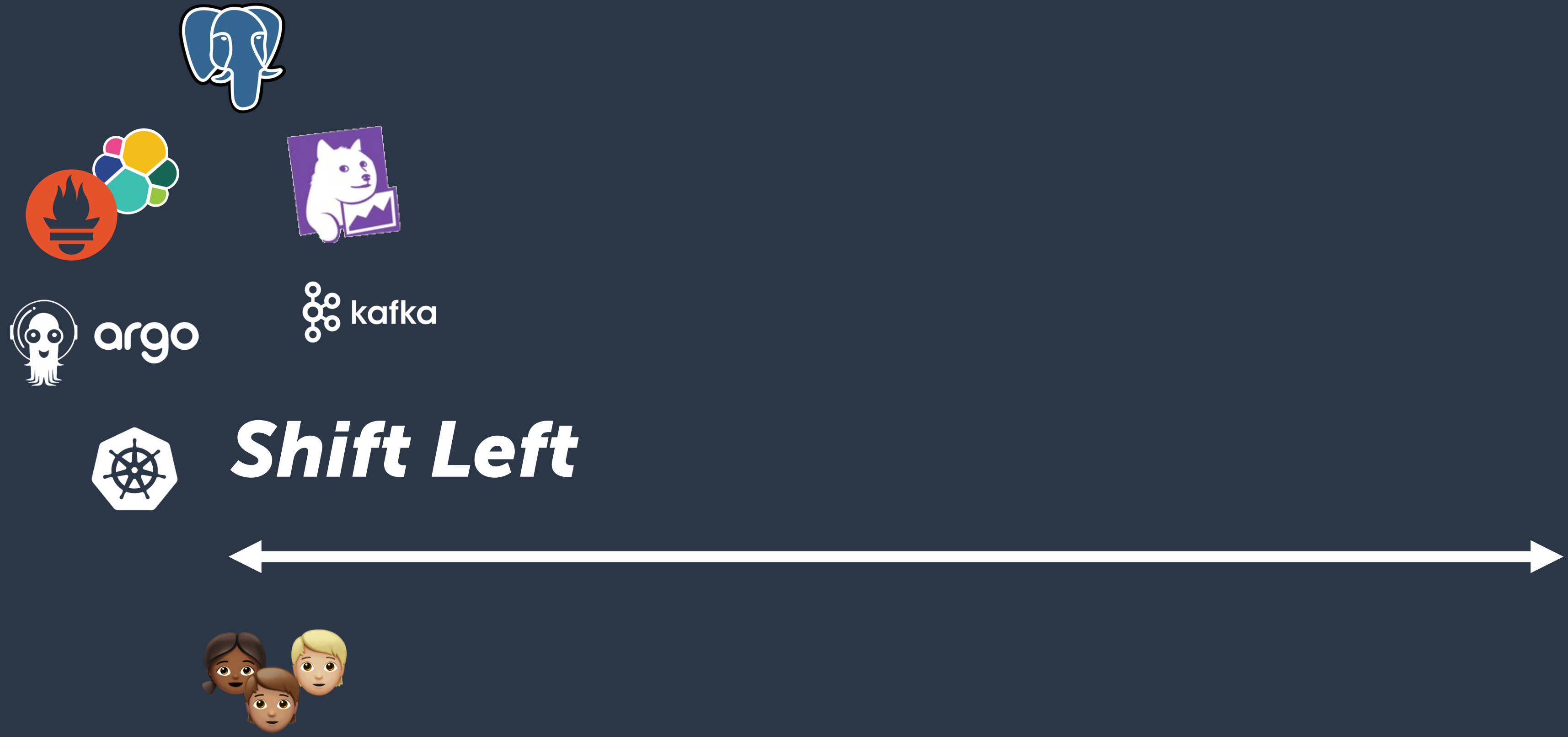
Which has heavily increased mental load within the engineering teams.

@duffleit


KV
Graduated
Vitess
CNCF Graduated
cloudevents
CNCF Incubating
NATS
CNCF Incubating
HELM
CNCF Graduated
Buildpacks.io
CNCF Incubating
OPERATOR FRAMEWORK
CNCF Incubating
argo
CNCF Incubating
flux
CNCF Incubating
heduling & Orchestration
Coordination & Service Discovery
Remote Procedure Call
Service Proxy
API Gateway
ernetes
Graduated
Crossplane
CNCF Incubating
CoreDNS
CNCF Graduated
etcd
CNCF Graduated
gRPC
CNCF Incubating
envoy
CNCF Graduated
CONTOUR
CNCF Incubating
EMISSARY INGRESS
CNCF Incubating
LINKE
CNCF Gra
Cloud Native Storage
Container Runtime
Cloud Nativ
OOK
Graduated
LONGHORN
CNCF Incubating
containerd
CNCF Graduated
cri-o
CNCF Incubating
cilium
CNCF Incubating
CNI
CNCF Incubating


kafka
argo

Shift Left



SQUER

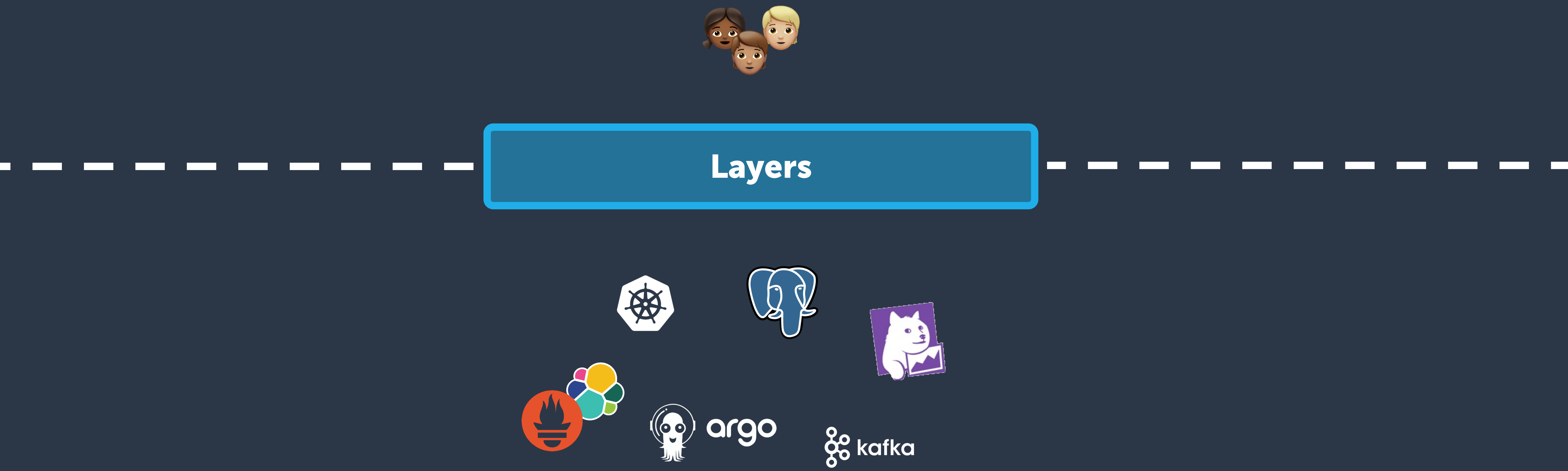
PILE LEFT ©

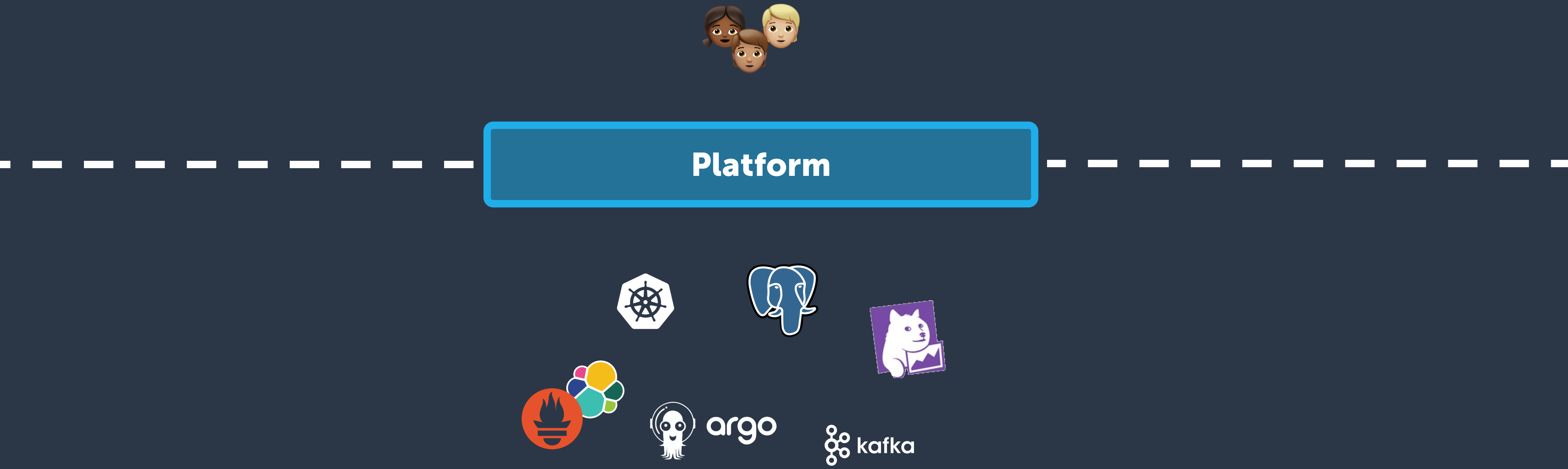
SQUER
kafka
argo
PILE LEFT ©
THIS IS FINE
@duffleit

PILE LEFT ©

We can solve any problem **by introducing an additional layer**.

— fundamental theorem of software engineering (FTSE)




Platform
argo
kafka

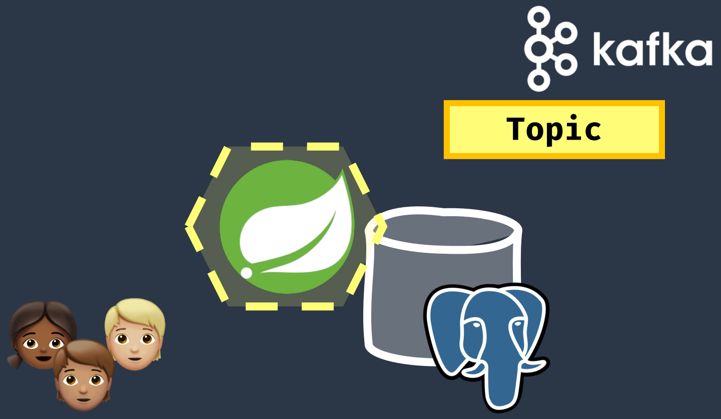
Platform

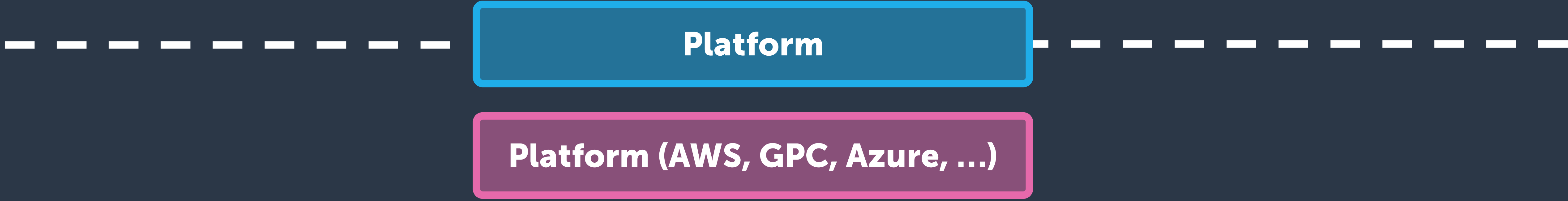
Platform (AWS, GPC, Azure, ...)

argo

kafka

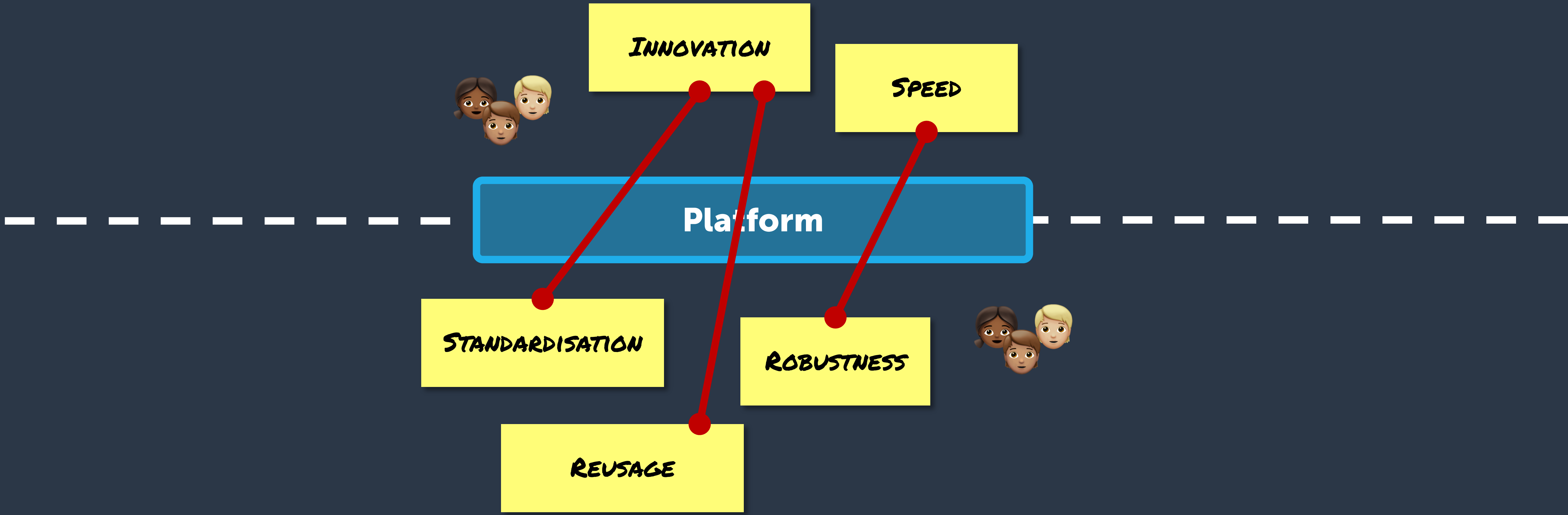
SQUER





@duffleit

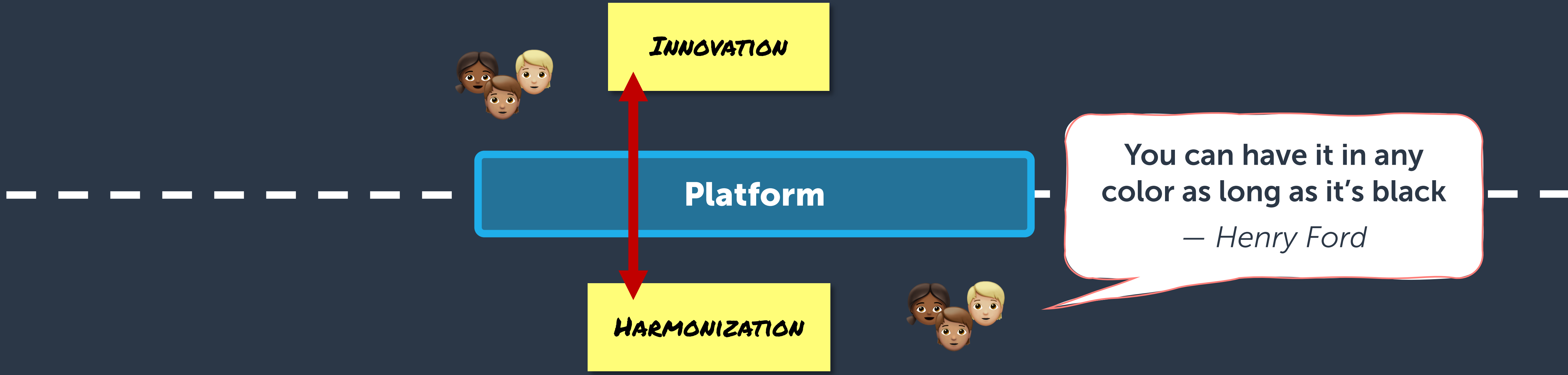
SQUER



@duffleit

Good Platforms enable Innovation by Harmonization.

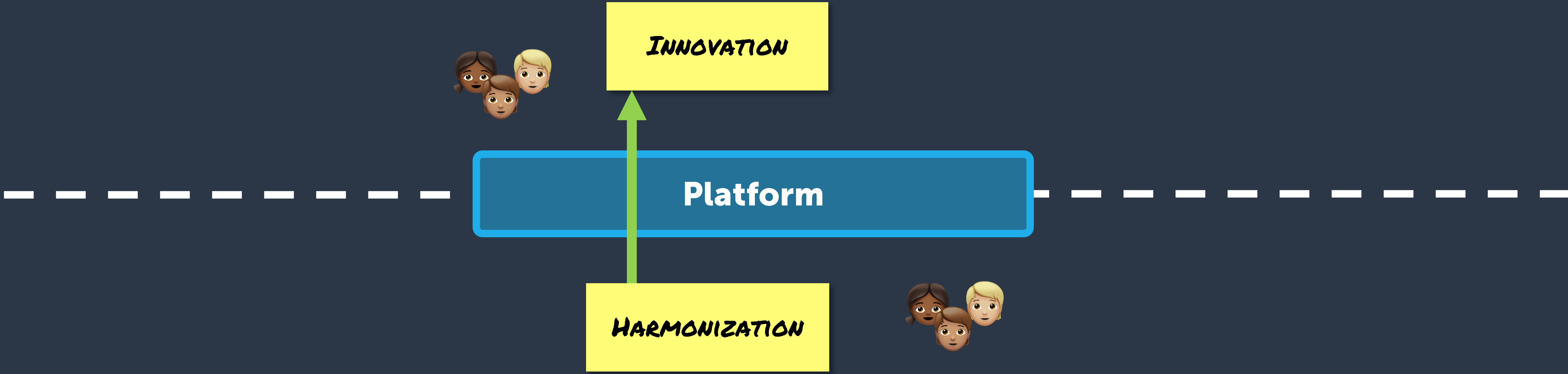
— Gregor Hohpe



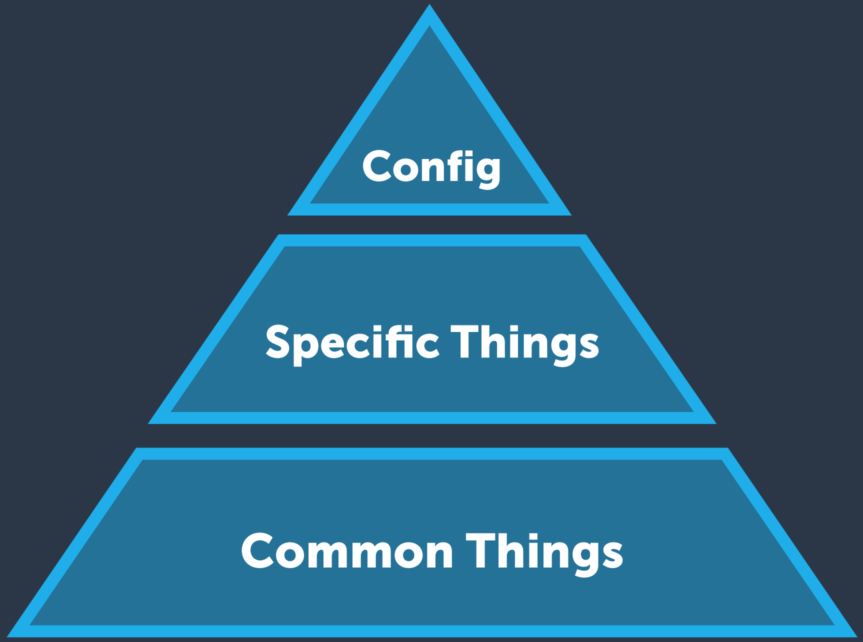
SQUER

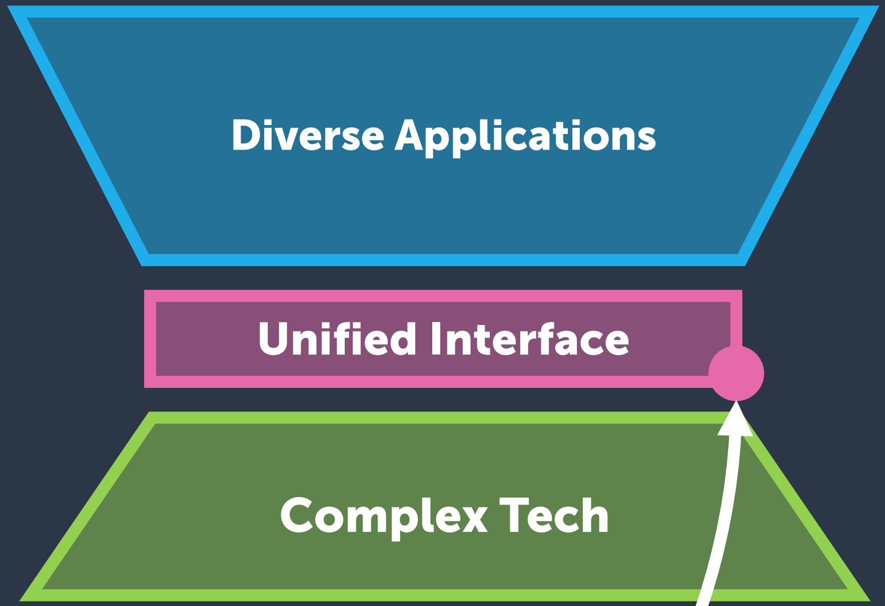
Good Platforms enable
Innovation by Harmonization.

— Gregor Hohpe



@duffleit





Platforms enable diverse Applications on unified interfaces.

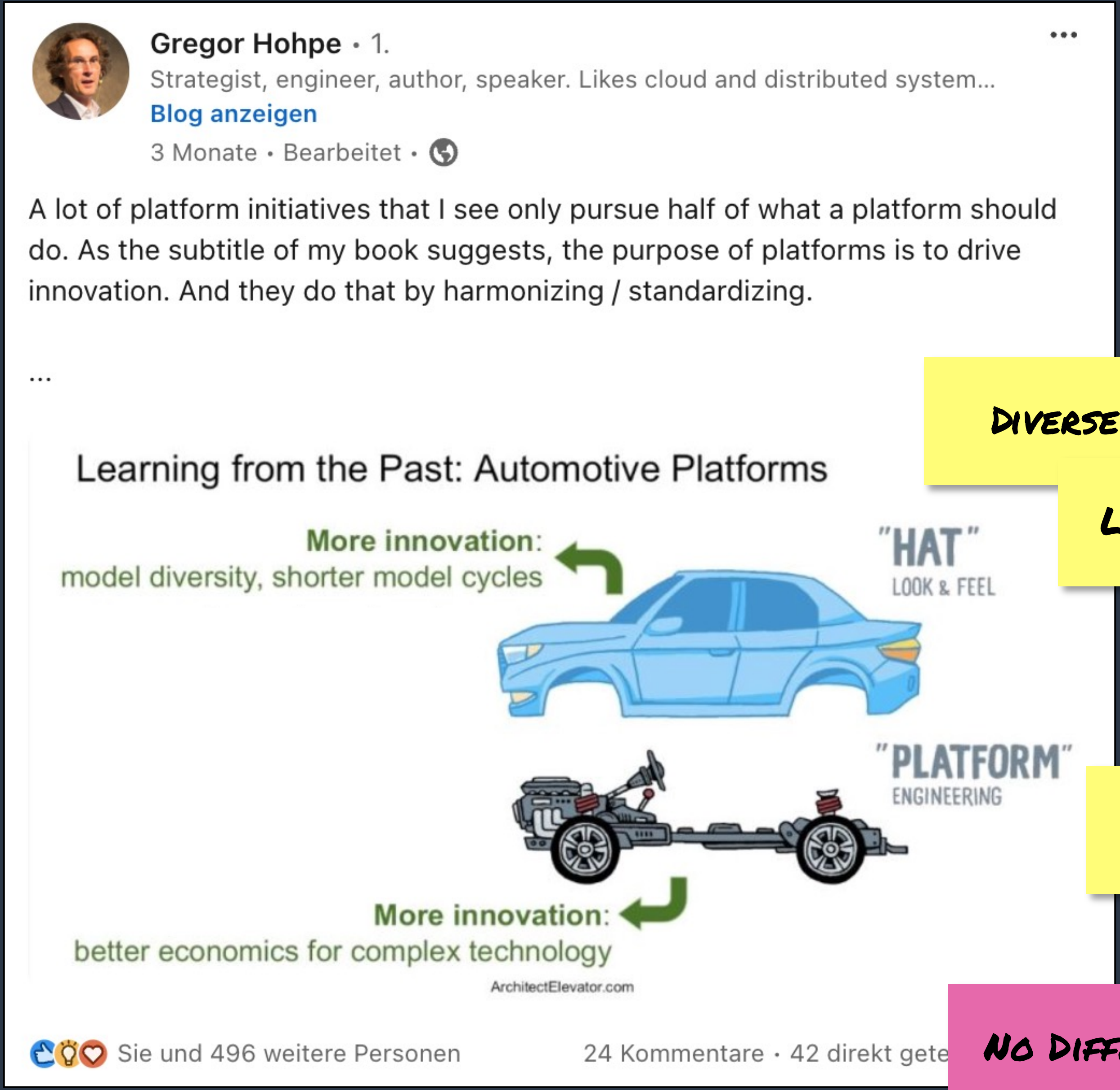
@duffleit

SQUER

If your users haven't built anything that surprises you, then you probably haven't developed a good platform.

— *Gregor Hohpe*

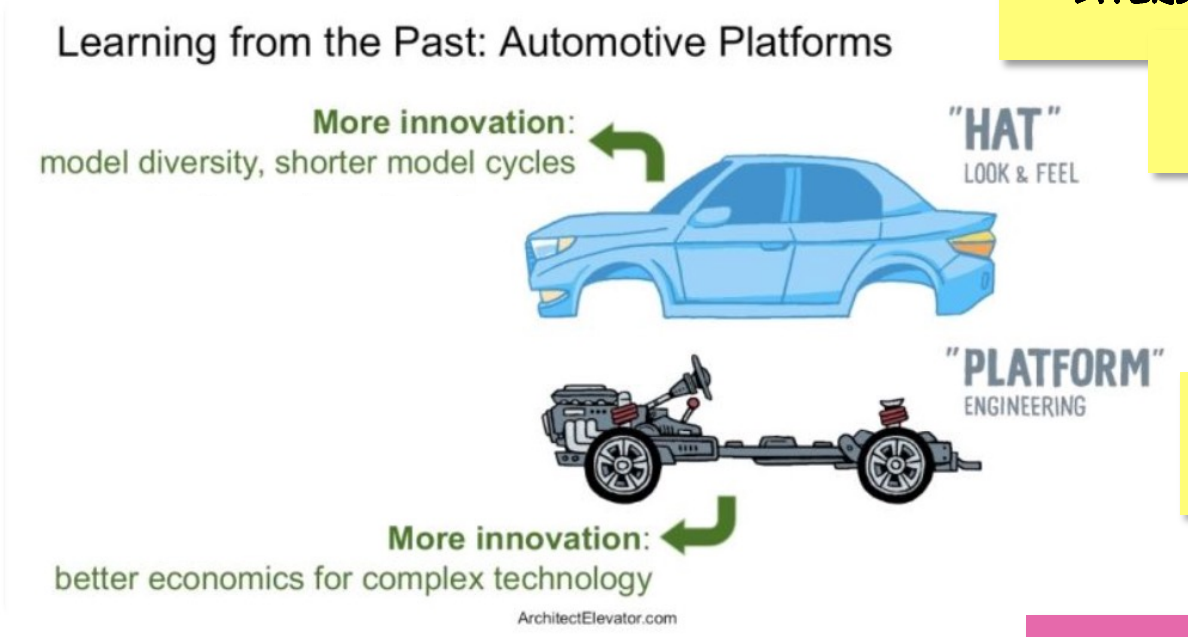
SQUER



Gregor Hohpe • 1.
Strategist, engineer, author, speaker. Likes cloud and distributed system...
Blog anzeigen
3 Monate • Bearbeitet •

A lot of platform initiatives that I see only pursue half of what a platform should do. As the subtitle of my book suggests, the purpose of platforms is to drive innovation. And they do that by harmonizing / standardizing.

...


Learning from the Past: Automotive Platforms
More innovation:
model diversity, shorter model cycles
"HAT"
LOOK & FEEL
"PLATFORM"
ENGINEERING
More innovation:
better economics for complex technology
ArchitectElevator.com

Sie und 496 weitere Personen
24 Kommentare • 42 direkt gete

DIVERSE

LOWER INVESMENT

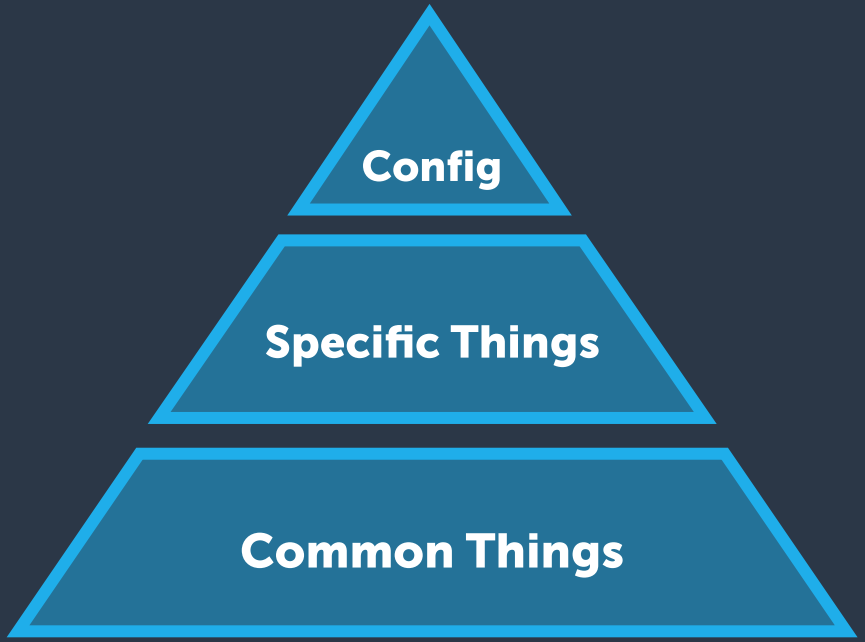
DIFFERENTIATOR

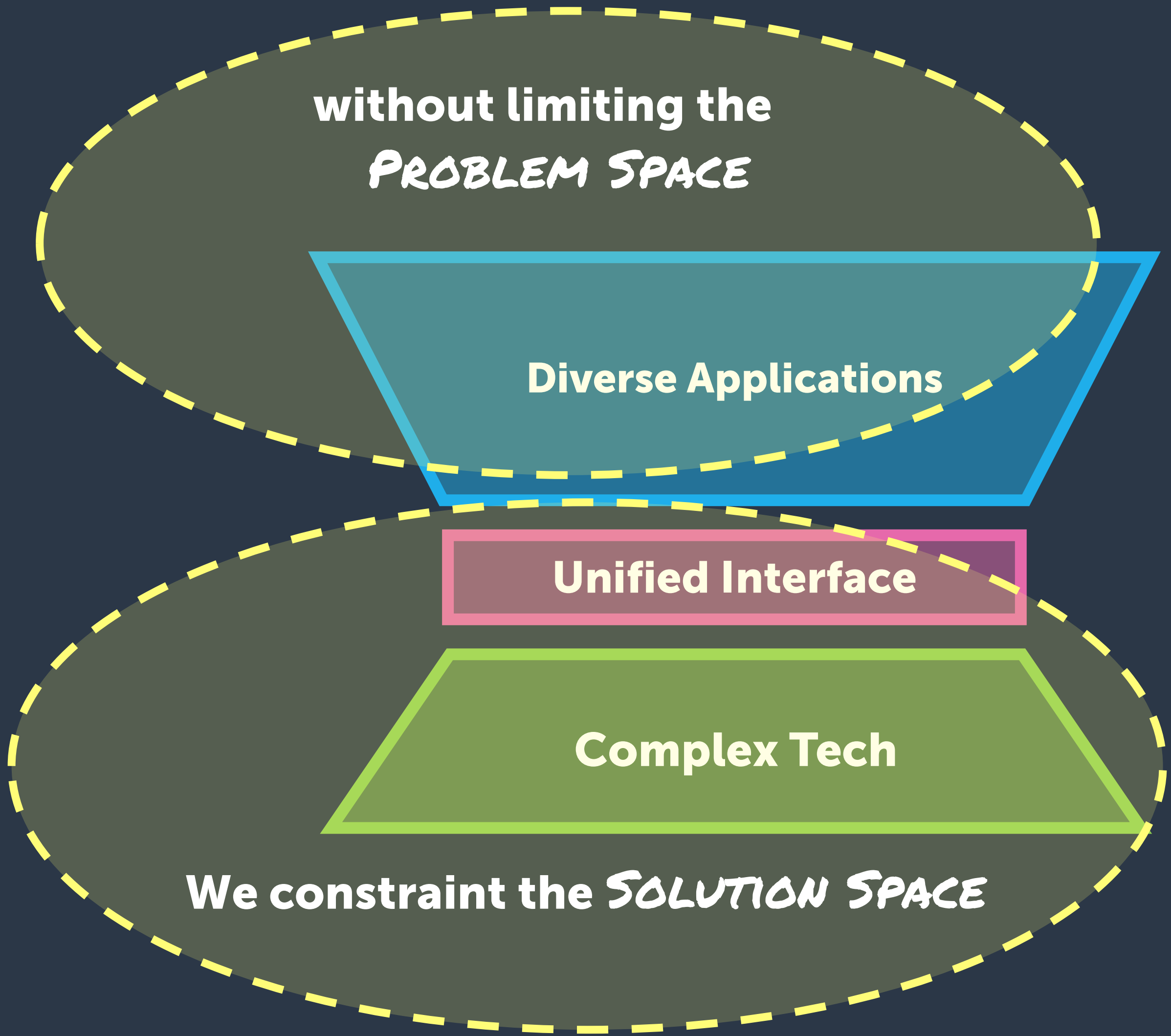
HIGH INVESMENT

STANDARDIZATION

NO DIFFERENTIATOR

@duffleit

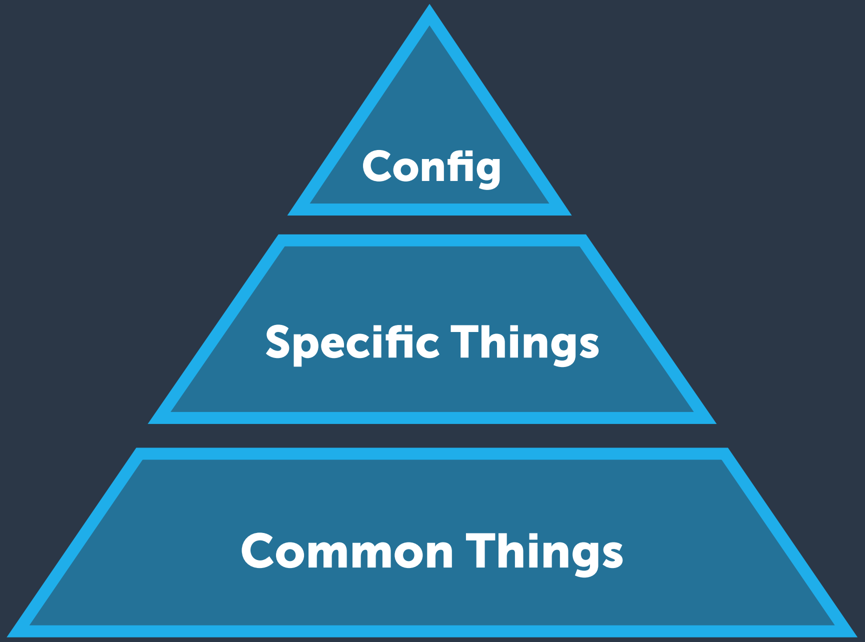


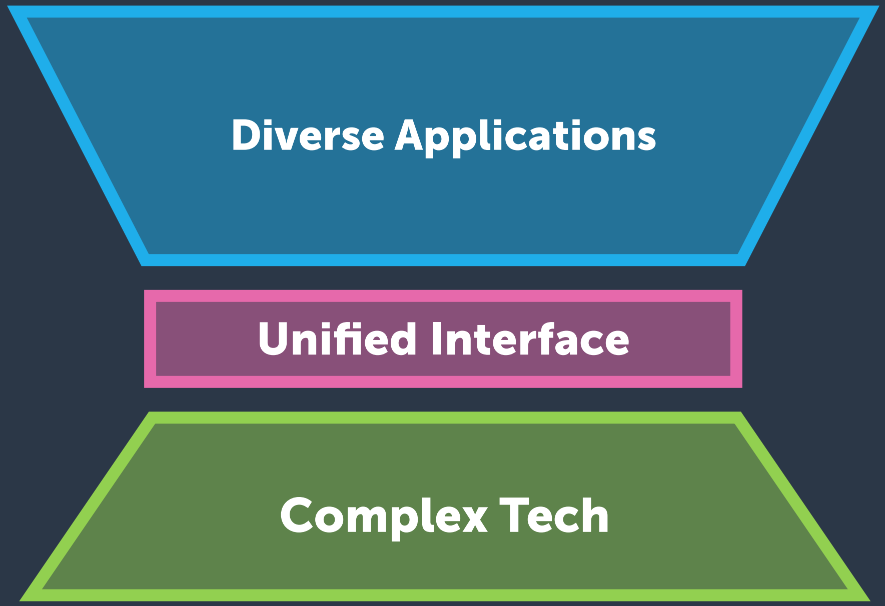


Platforms enable diverse Applications on unified interfaces.

@duffleit

SQUER





Economies of Scale.

Economies of Speed.

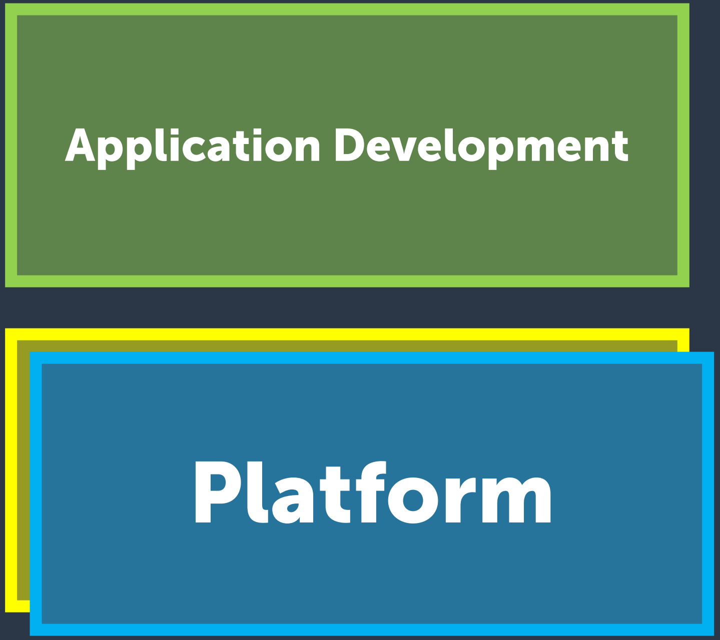
@duffleit

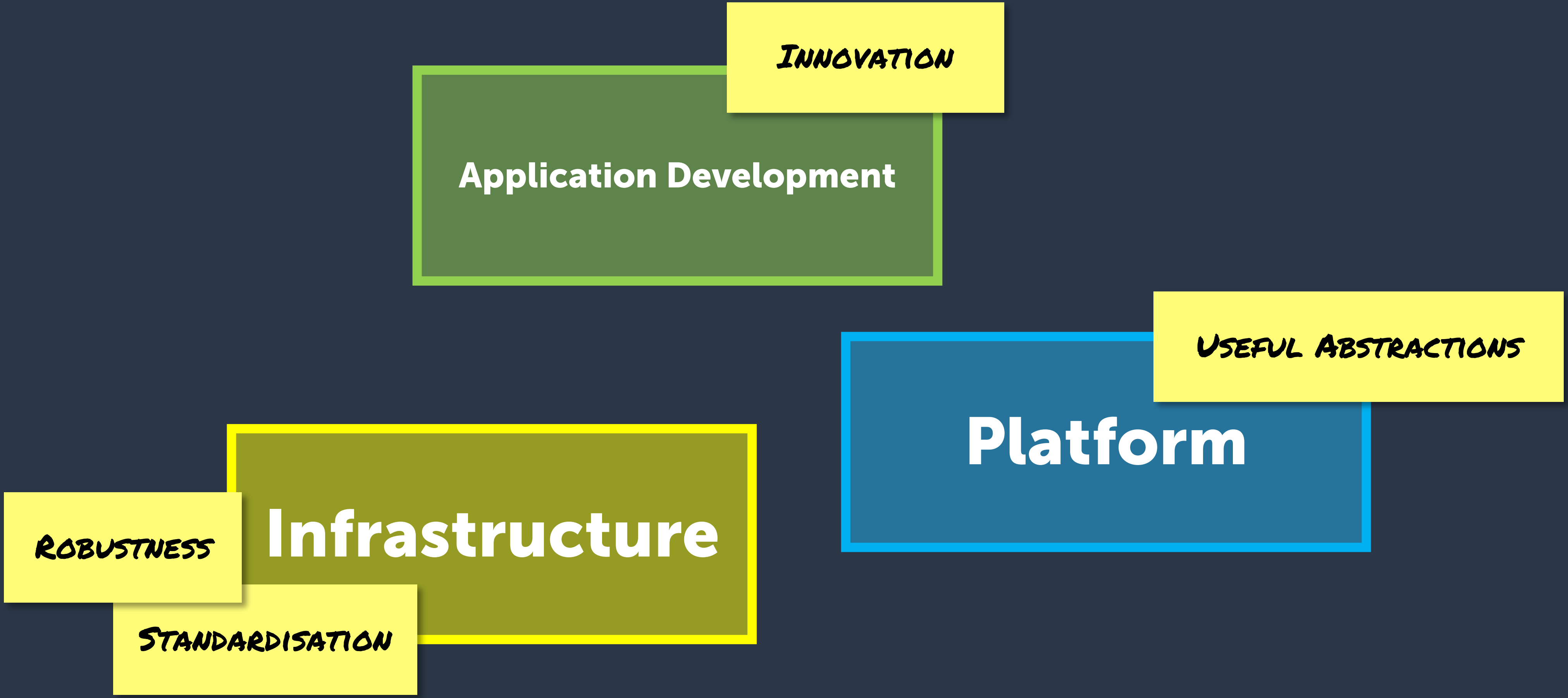
Platforms focus on accelerating the speed of delivery rather than on reuse and standardization.

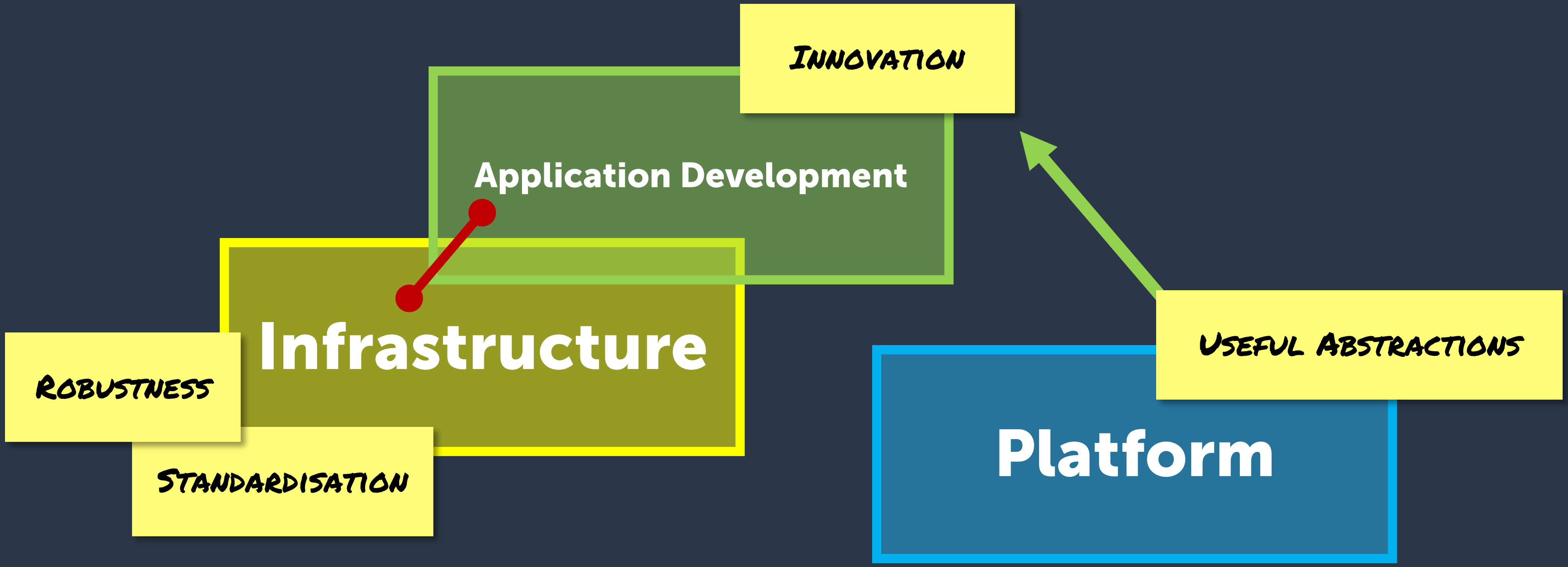
SQUER

Application Development
Infrastructure

@duffleit


Application Development
Platform


Innovation
Application Development
Useful Abstractions
Platform
Infrastructure
Robustness
Standardisation



SQUER

@duffleit

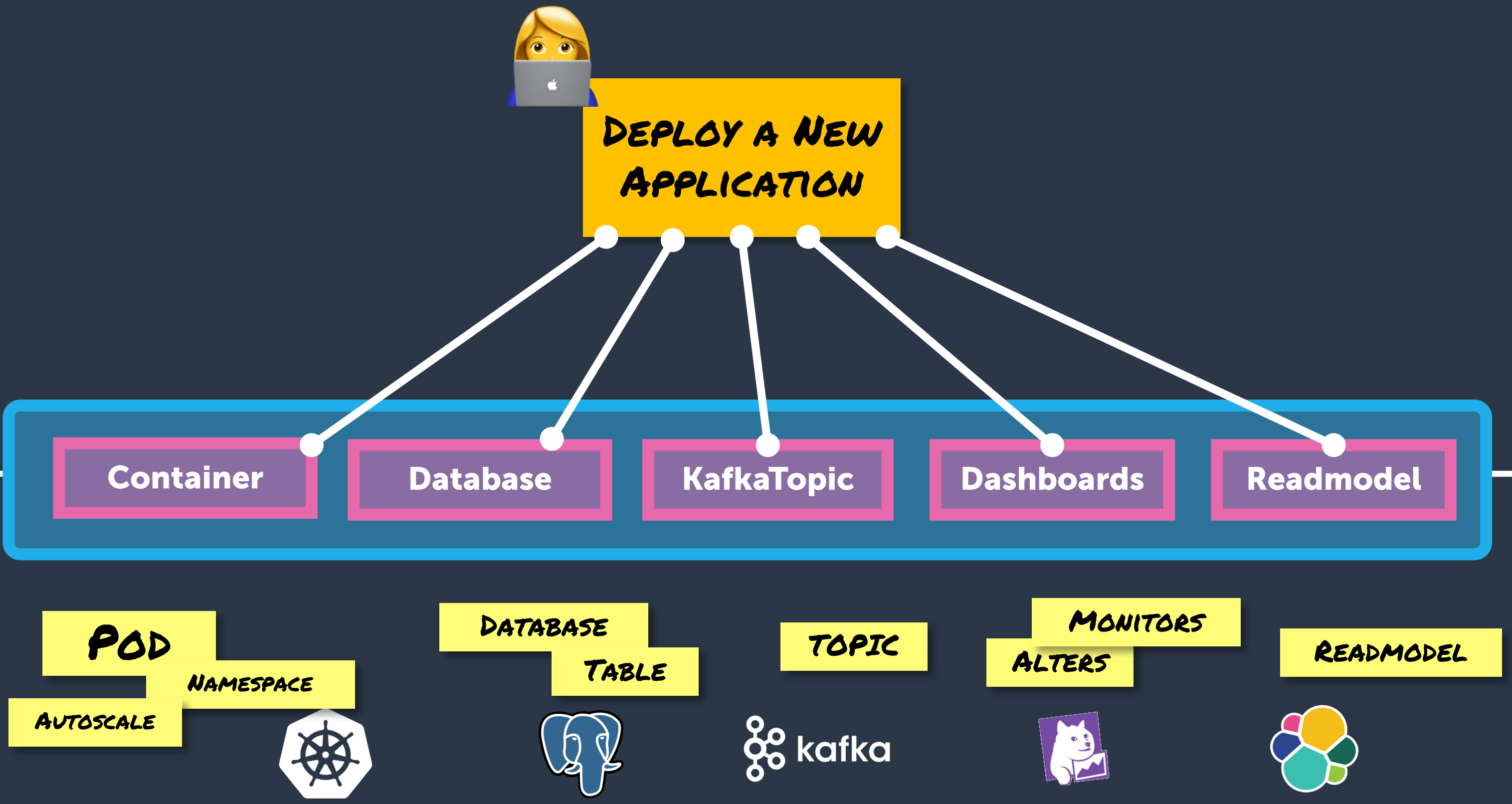
SQUER

We do this by useful

Abstractions

@duffleit

SQUER

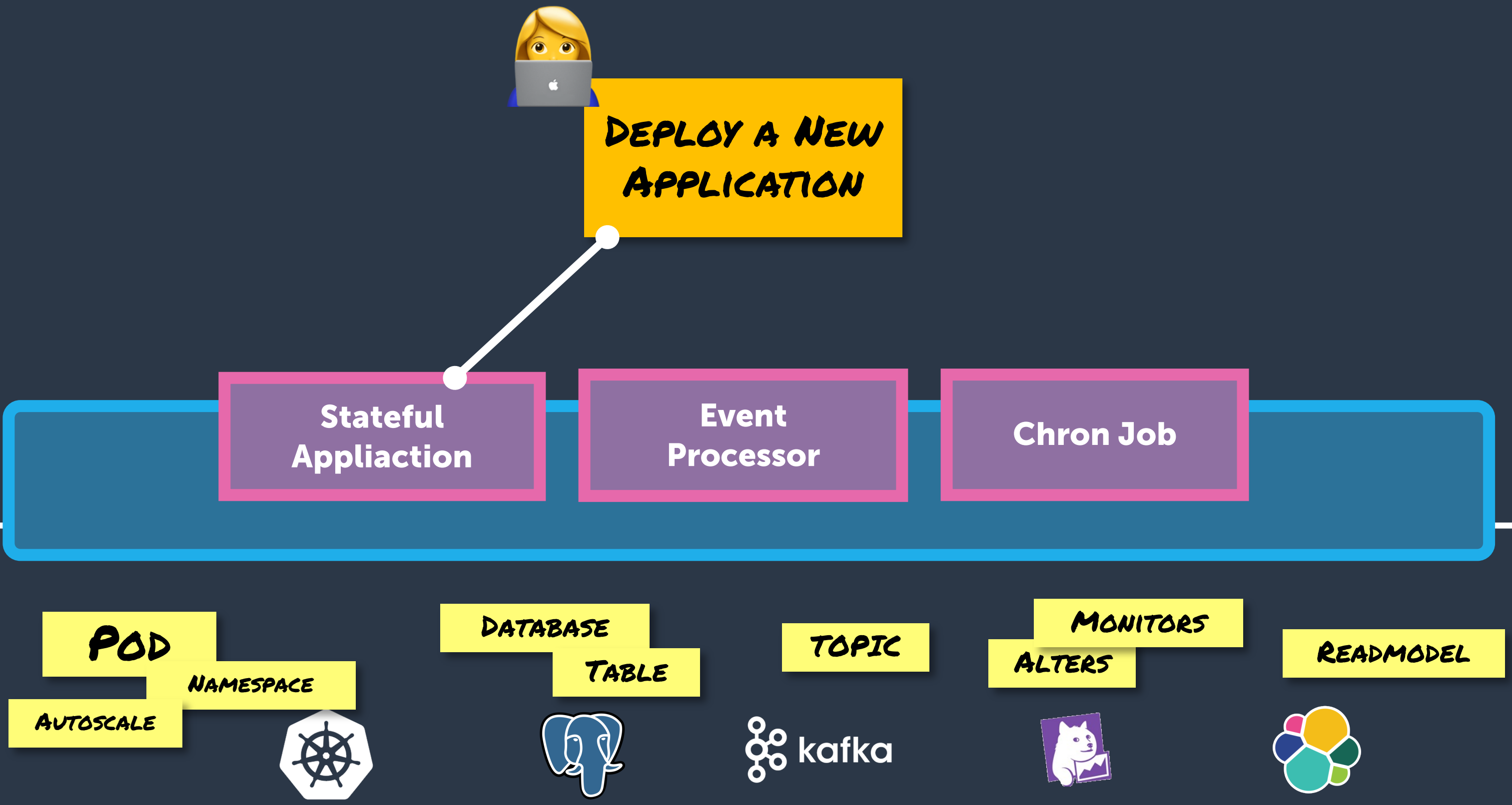


Infrastructure Centric View,
based on Composition.

@duffleit

SQUER

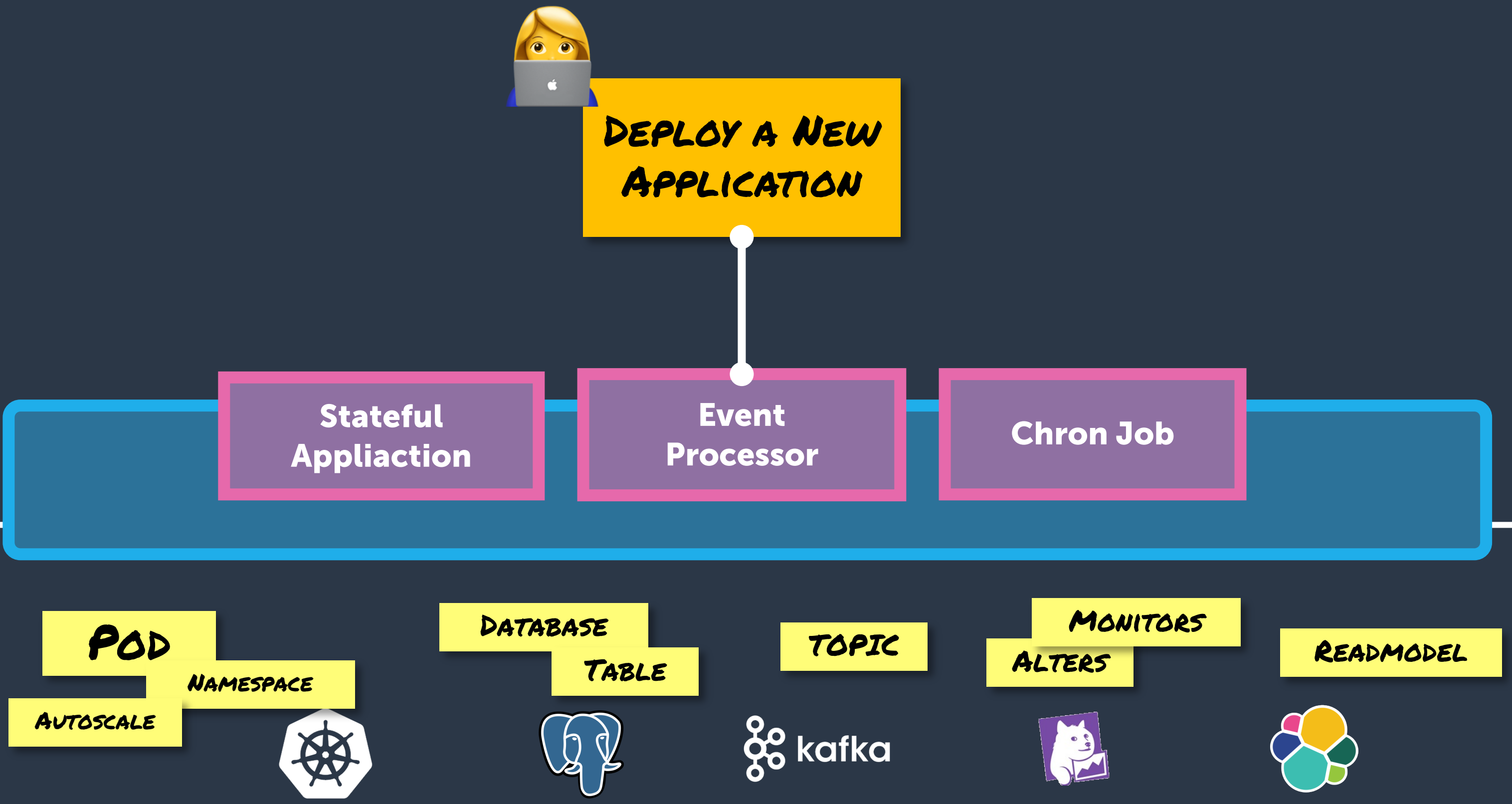
Developer Centric View,
based on Abstractions.



@duffleit

SQUER

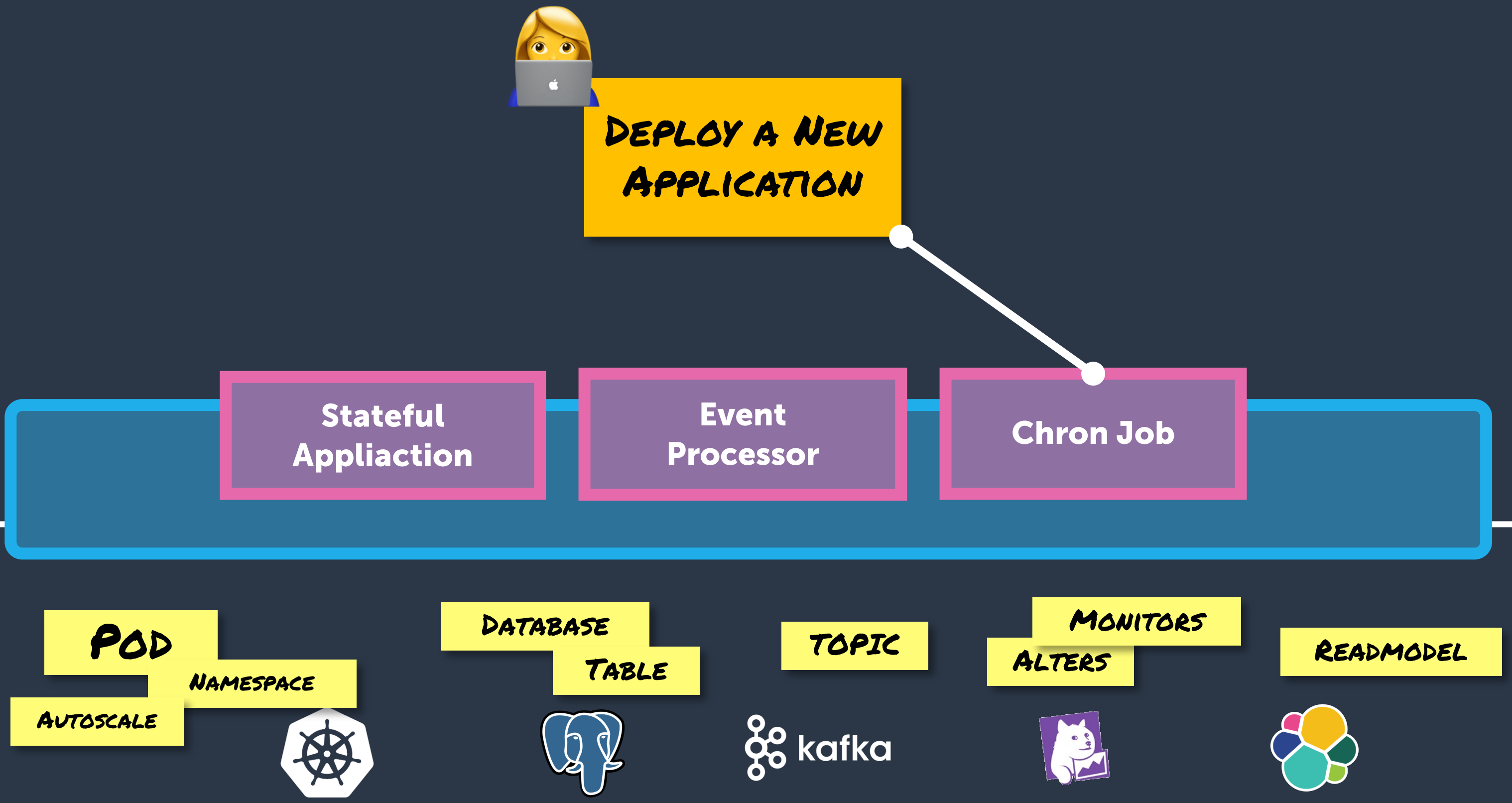
Developer Centric View,
based on Abstractions.



@duffleit

SQUER

Developer Centric View,
based on Abstractions.



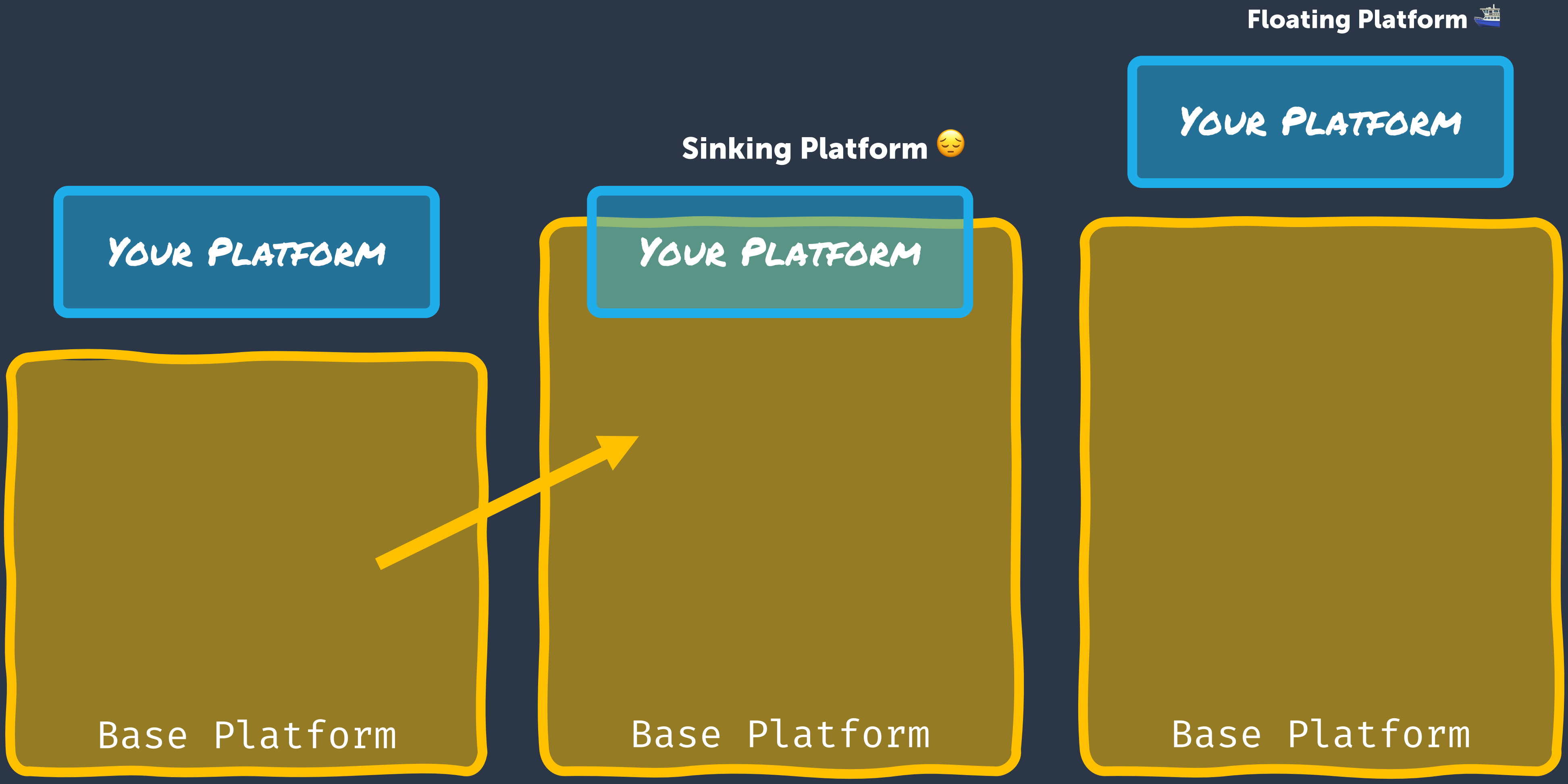
@duffleit

SQUER

Platform Interfaces are mainly about Abstractions and less about Composition.

@duffleit

SQUER

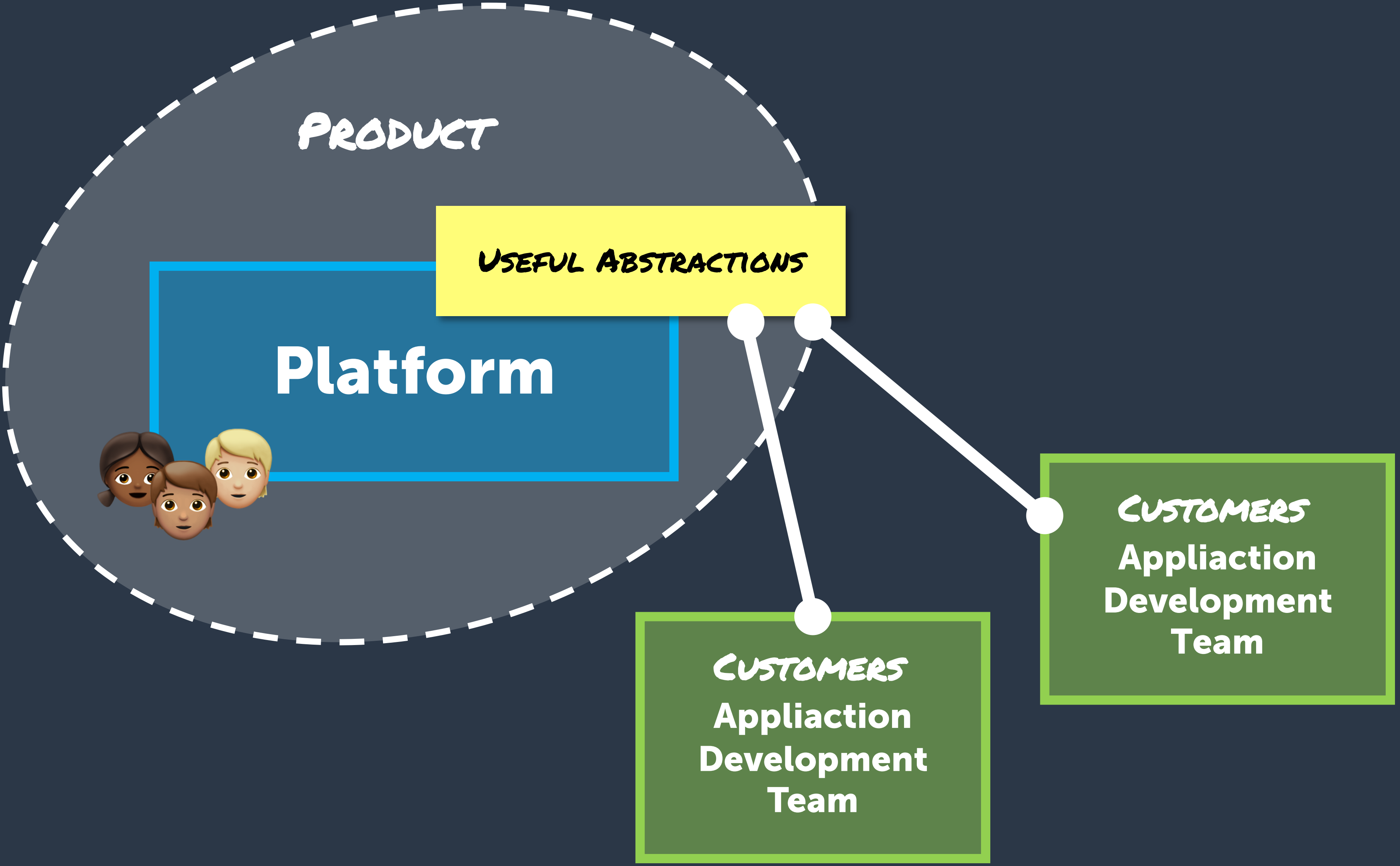


@duffleit

SQUER

Platform Interfaces are mainly about Abstractions and less about Composition.

@duffleit


Product
Useful Abstractions
Platform
Customers
Appliaction Development Team
Customers
Appliaction Development Team

@duffleit

SQUER

Consider your Platform as a Product, and staff it like that.

@duffleit

SQUER

Ok,

Let’s sum up.

@duffleit

SQUER

When autonomous teams are your organizational scaling strategy,

@duffleit

SQUER

When autonomous teams* are your organizational scaling strategy, then mental load within those teams becomes your biggest obstacle

@duffleit

SQUER

When autonomous teams are your organizational scaling strategy, then mental load within those teams becomes your biggest obstacle, and platform engineering your most promising ally in overcoming it.

🚀 Platform Engineering is focused on accelerating the speed of delivery rather than increasing reuse and standardization.

💅 It is not simply about rebranding your existing infrastructure teams, as they address different problems.

🧿 Great platforms define APIs based on useful abstractions rather than leaky compositions.

🧑‍🤝‍🧑 Treat and measure your platform teams as product teams, and therefore staff them accordingly.

SQUER

DAVID LEITNER

Principal Engineer

SQUER

👋 david@squer.io

🌎 @duffleit

Slides

@duffleit