nine ways to fail at cloud native

Holly Cummins
IBM Garage
@holly_cummins

IBM

I’m a consultant with the IBM Garage.

These are my scary stories

fail

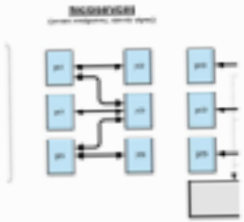
the magic morphing meaning

what **is** cloud native?

Daniel Bryant
@danielbryantuk

Following

I've gotta hand it to @bibryam, he's got a great way of framing things... :-)

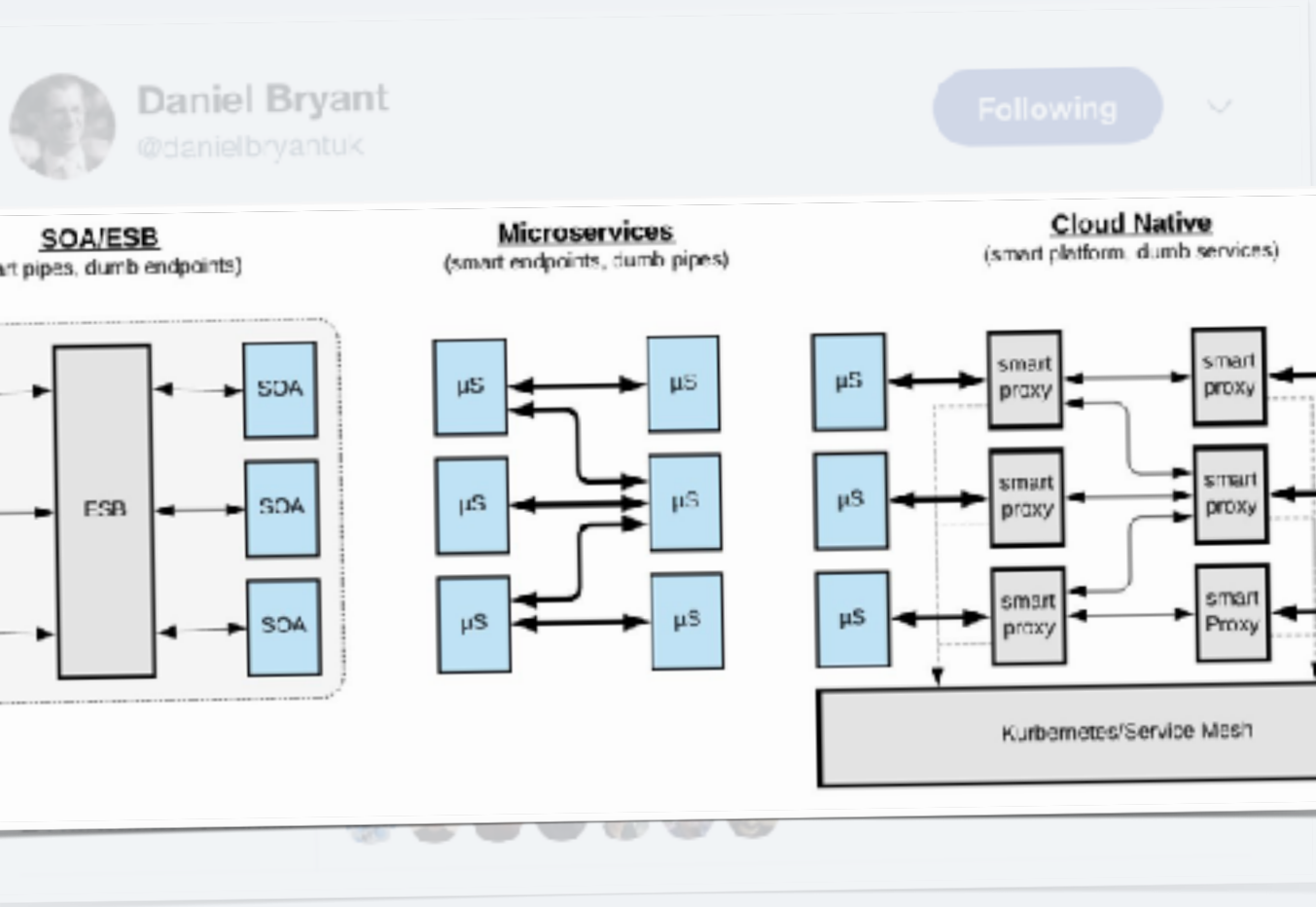


Bilgin Ibryam @bibryam
ESB -> Microservices -> CloudNative

8:42 AM - 7 Aug 2018

2 Retweets **7** Likes

Credit to @bibryam



#IBMGarage

@holly_cummins

(a great article, btw)

InfoQ

Development | Architecture & Design | AI, ML & Data Engineering | Culture & Methods | DevOps | Videos with Transcripts

Streaming | Machine Learning | Reactive | Microservices | Containers | .NET | All topics | The Architects' Newsletter

You are here: InfoQ Homepage » Articles » Microservices in a Post-Kubernetes Era

Microservices in a Post-Kubernetes Era

Posted by Bilgin Ibryam, reviewed by Daniel Bryant on Sep 01, 2018. Estimated reading time: 8 minutes

Key Takeaways

- The microservice architecture is still the most popular architectural style for distributed systems. But Kubernetes and the cloud native movement has redefined certain aspects of application design and development at scale.
- On a cloud native platform, observability of services is not enough. A more fundamental prerequisite is to make microservices automatable, by implementing health checks, reacting to signals, declaring resource consumption, etc.
- In the post-Kubernetes era, using libraries to implement operational networking

RELATED CONTENT

eBay Replatforming to Kubernetes, Envoy and Kafka: Intending to Open Source Hardware and Software

Networking Your Microservices Applications

Designing Events-First Microservices

#IBMGarage

(a great article, btw)

InfoQ

Development | Architecture & Design | AI, ML & Data Engineering | Culture & Methods | DevOps | Videos with Transcripts

Streaming | Machine Learning | Reactive | Microservices | Containers | .NET | All topics

The Architects' Newsletter

Microservices

8 minutes | 1 Discuss

Share

Reading List | Read later

Key Takeaways

- The microservice architecture is still the most popular architectural style for distributed systems. But Kubernetes and the cloud native movement has redefined certain aspects of application design and development at scale.
- On a cloud native platform, observability of services is not enough. A more fundamental prerequisite is to make microservices automatable, by implementing health checks, reacting to signals, declaring resource consumption, etc.
- In the post-Kubernetes era, using libraries to implement operational networking

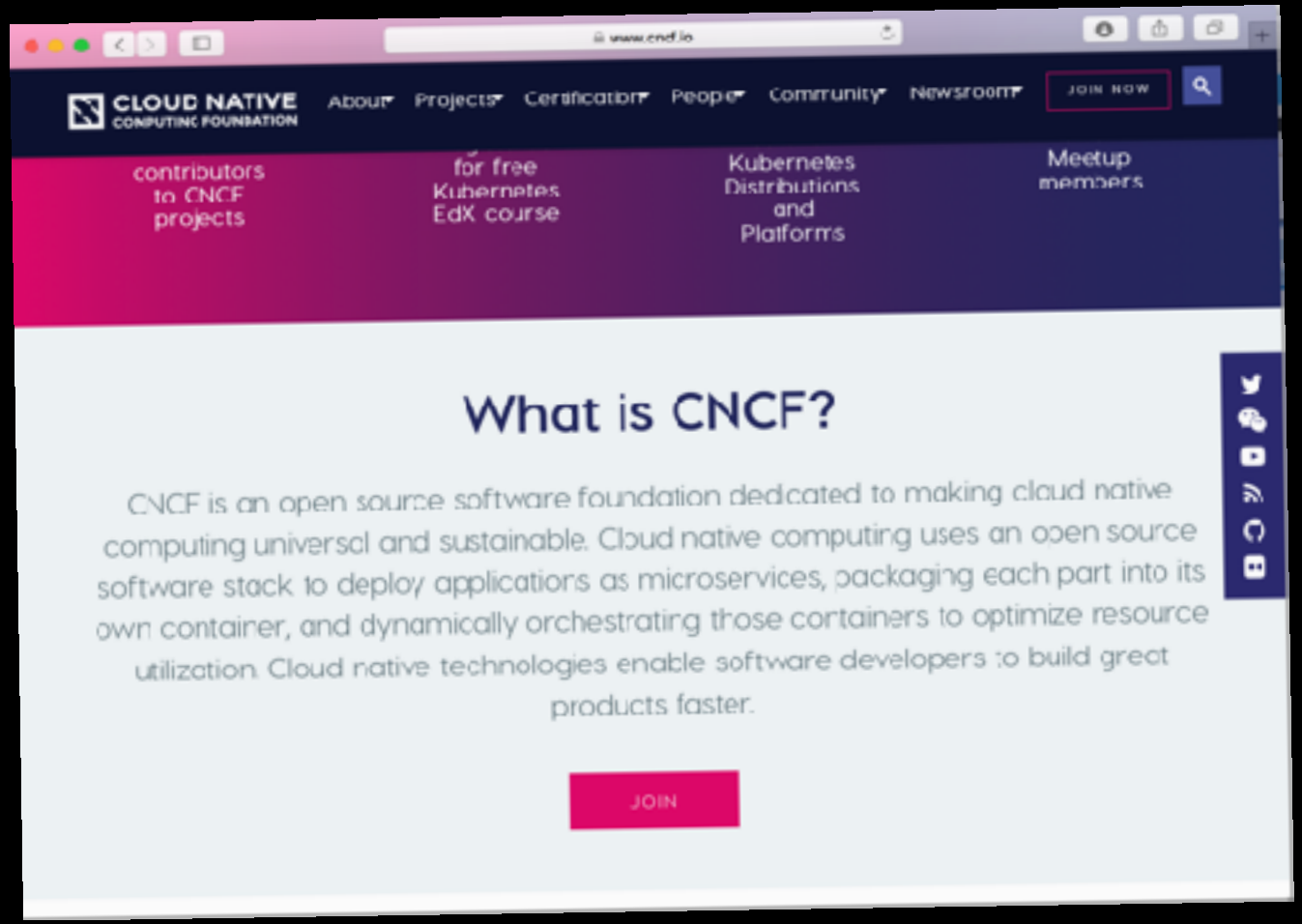
RELATED CONTENT

eBay Replatforming to Kubernetes, Envoy and Kafka: Intending to Open Source Hardware and Software Sep 18, 2018

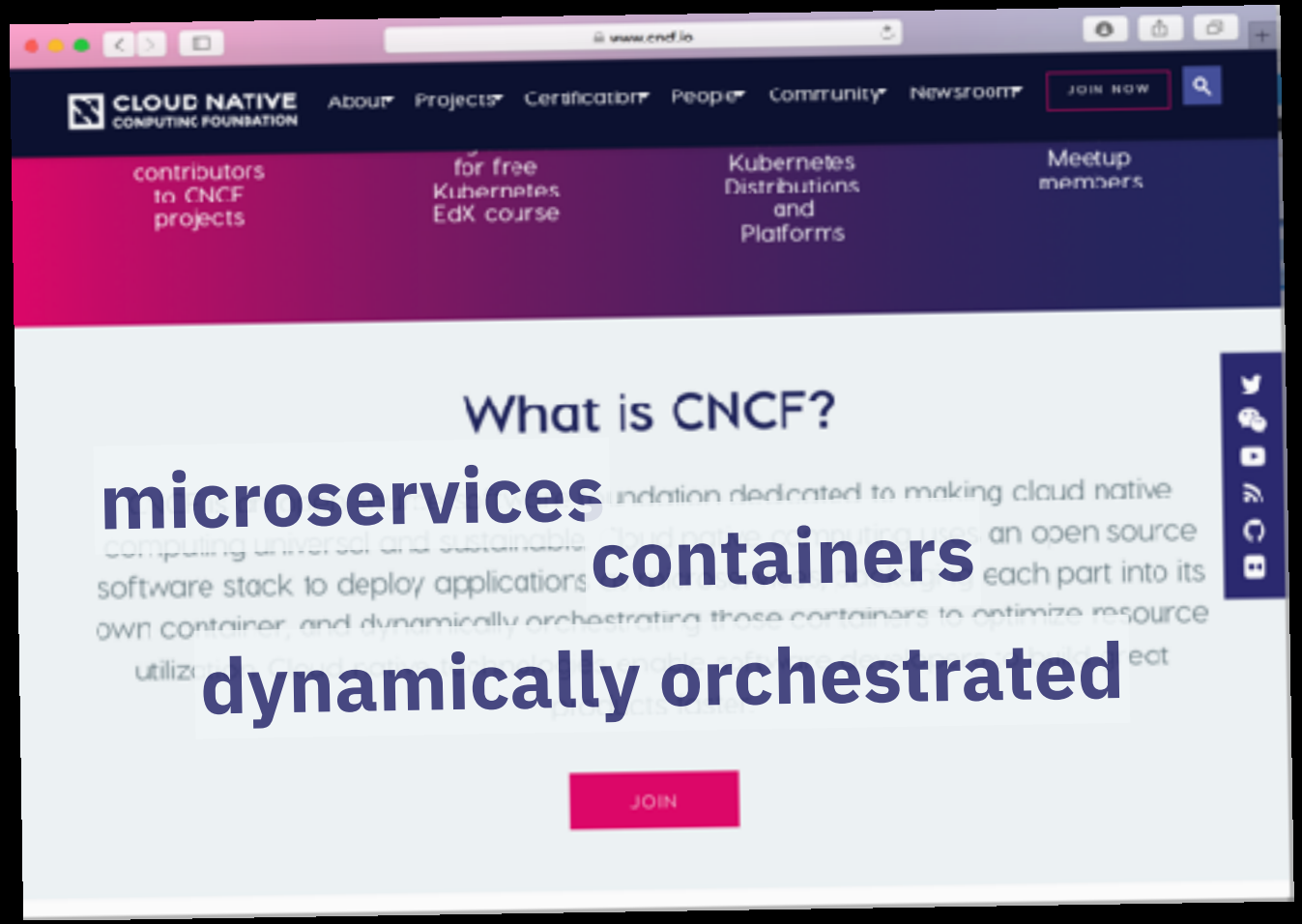
Networking Your Microservices Applications Sep 21, 2018

Designing Events-First Microservices Sep 13, 2018

#IBMGarage


www.cncf.io
CLOUD NATIVE
COMPUTING FOUNDATION
About
Projects
Certification
People
Community
Newsroom
JOIN NOW
contributors to CNCF projects
for free Kubernetes EdX course
Kubernetes Distributions and Platforms
Meetup members
What is CNCF?
CNCF is an open source software foundation dedicated to making cloud native computing universal and sustainable. Cloud native computing uses an open source software stack to deploy applications as microservices, packaging each part into its own container, and dynamically orchestrating those containers to optimize resource utilization. Cloud native technologies enable software developers to build great products faster.
JOIN

2019


www.cncf.io
CLOUD NATIVE
COMPUTING FOUNDATION
About
Projects
Certification
People
Community
Newsroom
JOIN NOW
contributors to CNCF projects
for free Kubernetes EdX course
Kubernetes Distributions and Platforms
Meetup members
What is CNCF?
microservices
containers
dynamically orchestrated
JOIN

2019

flowcon

"the cloud native computing foundation is **wrong** ... about cloud native."

Dr Holly

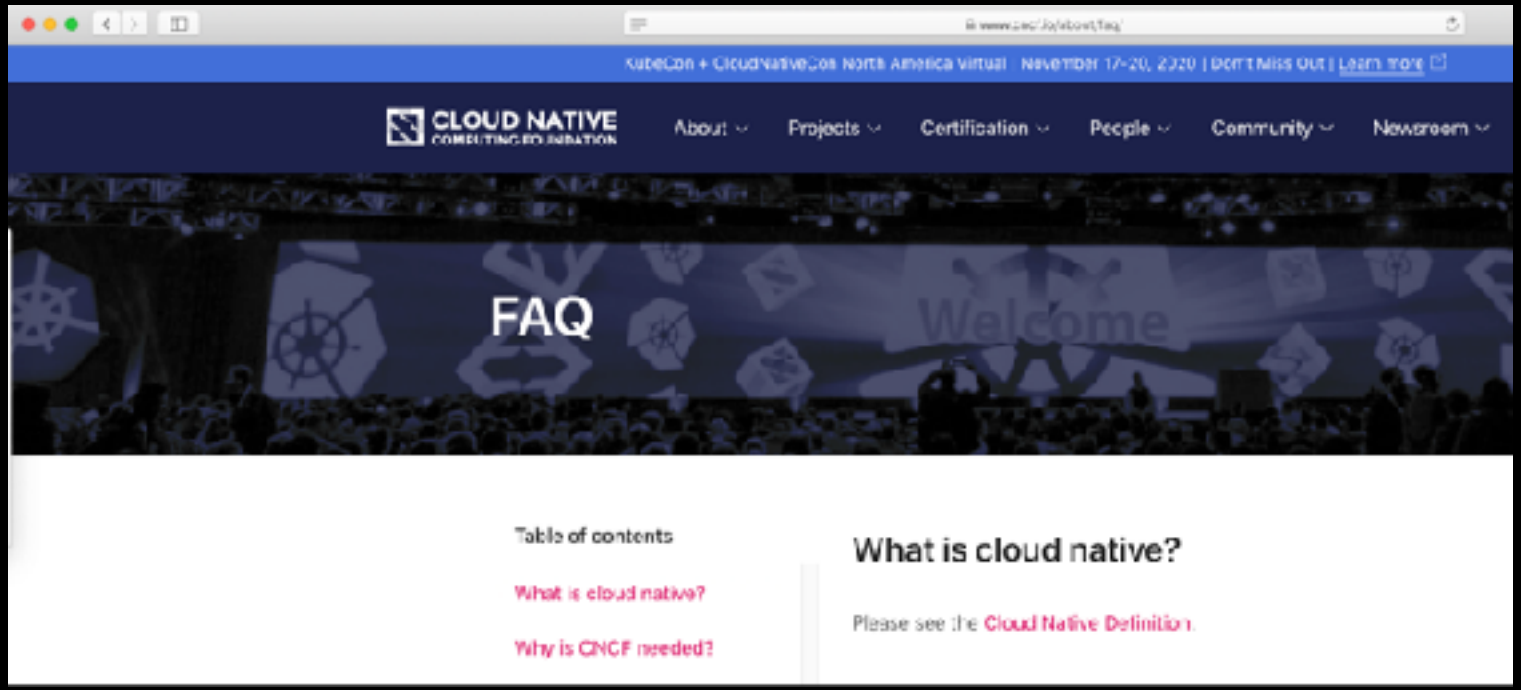
flowcon

"the cloud native computing foundation is **wrong** ... about cloud native."

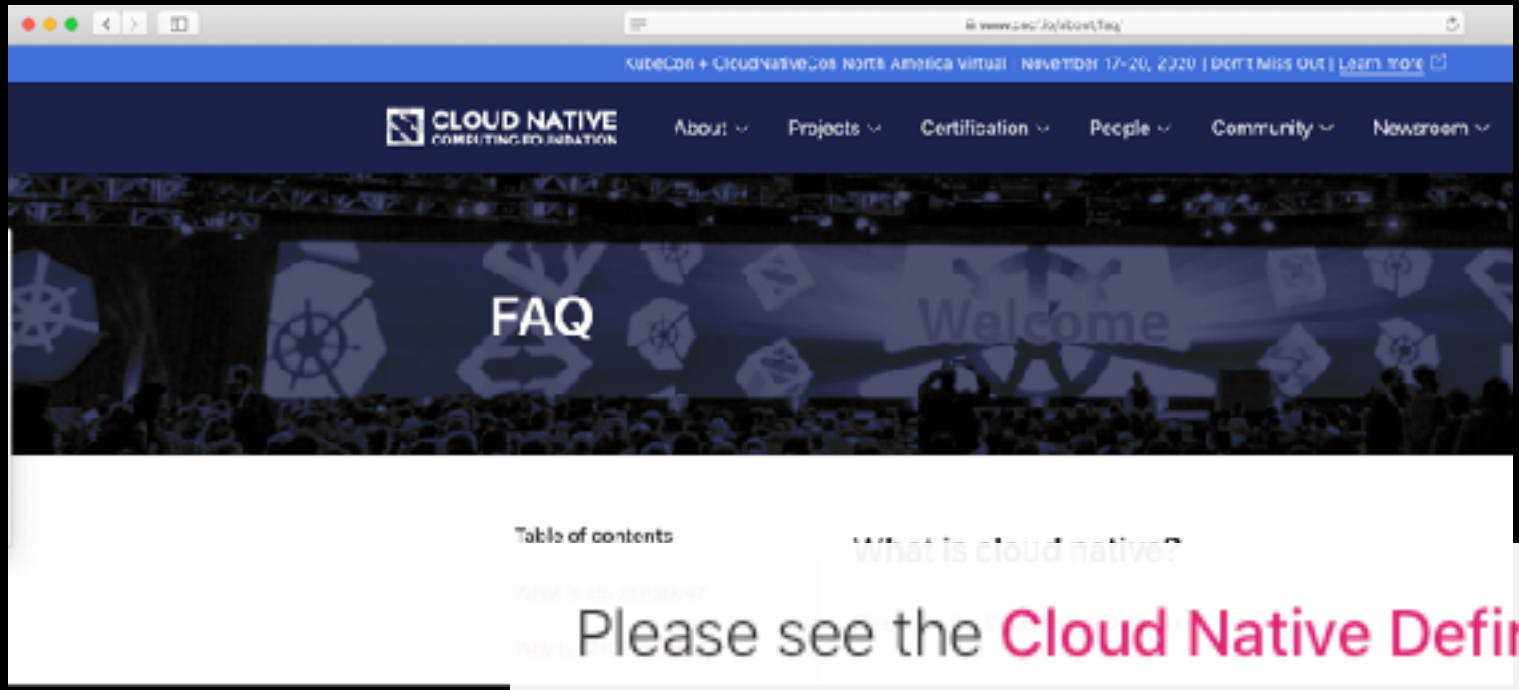
Dr Holly

Dr Holly


Sam Newman
@samnewman
Interesting side effect of the Cloud Native Foundation is now that I'm commonly speaking to people who believe it can't be cloud native without Kubernetes
10:38 PM · Nov 19, 2020 · Tweetbot for iOS
15 Retweets
4 Quote Tweets
130 Likes


KubeCon + CloudNativeCon North America Virtual November 17-20, 2020 | Don't Miss Out | Learn more
CLOUD NATIVE
About
Projects
Certification
People
Community
Newsroom
FAQ
Welcome
Table of contents
What is cloud native?
Why is CNCF needed?
What is cloud native?
Please see the Cloud Native Definition

2020



Please see the Cloud Native Definition.

2020

CNCF Cloud Native Definition v1.0

Approved by TOC: 2018-06-11

Français (French) | Polski (Polish) | Português Brasileiro (Portuguese) | Русский (Russian)

Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.

These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil.

The Cloud Native Computing Foundation seeks to drive adoption of this paradigm by fostering and sustaining an ecosystem of open source, vendor-neutral projects. We democratize state-of-the-art patterns to make these innovations accessible to everyone.

العربية

cncf / toc

toc / DEFINITION.md

18 contributors +6

CNCF Cloud Native Definition v1.0

Approved by TOC: 2018-06-11

العربية (Arabic) | 中文(简体) (Chinese) | 日本語 (Japanese) | 한국어 (Korean) | Deutsch (German) | Español (Spanish)
Français (French) | Polski (Polish) | Português Brasileiro (Portuguese) | Русский (Russian)

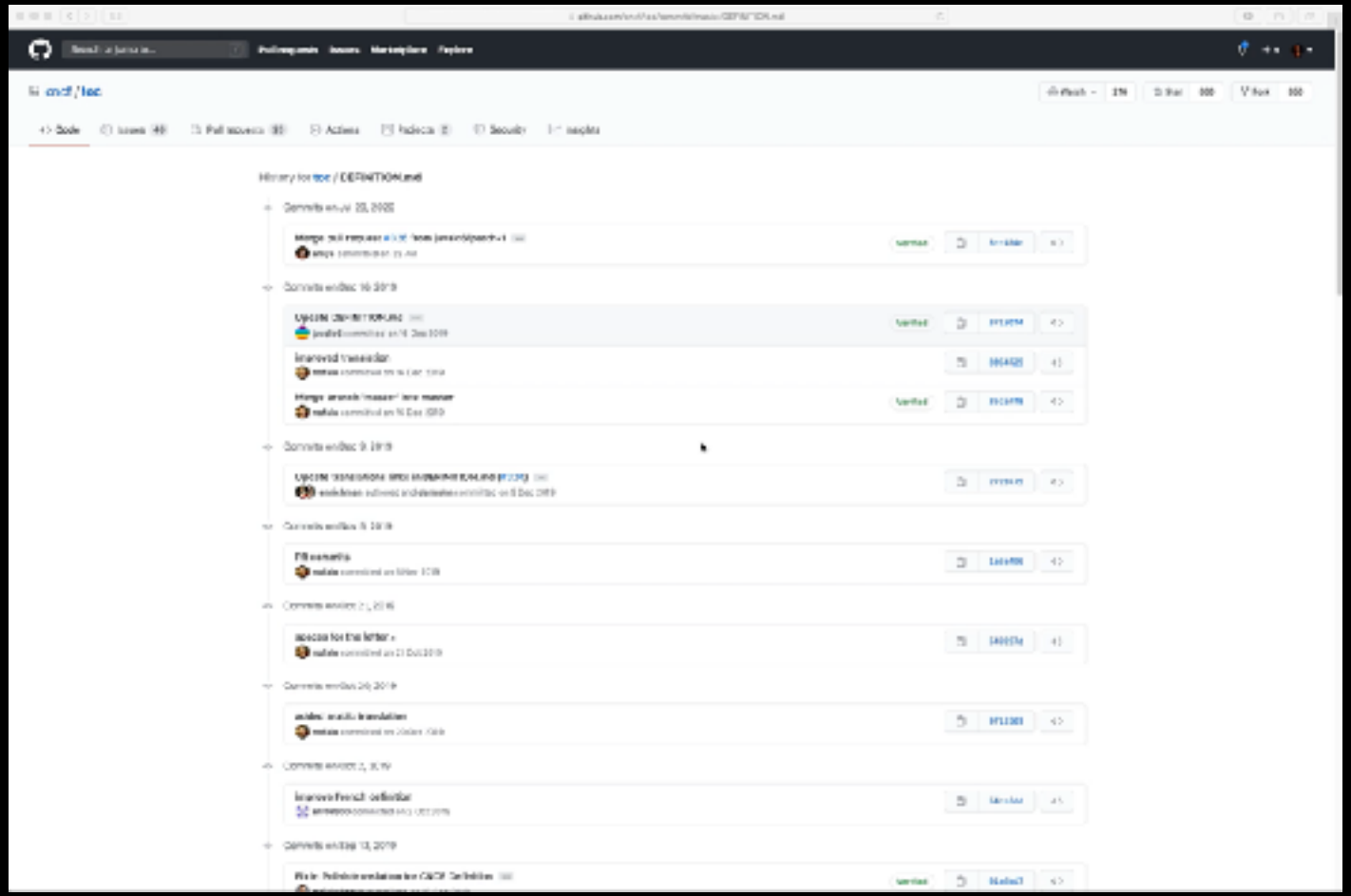
Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.

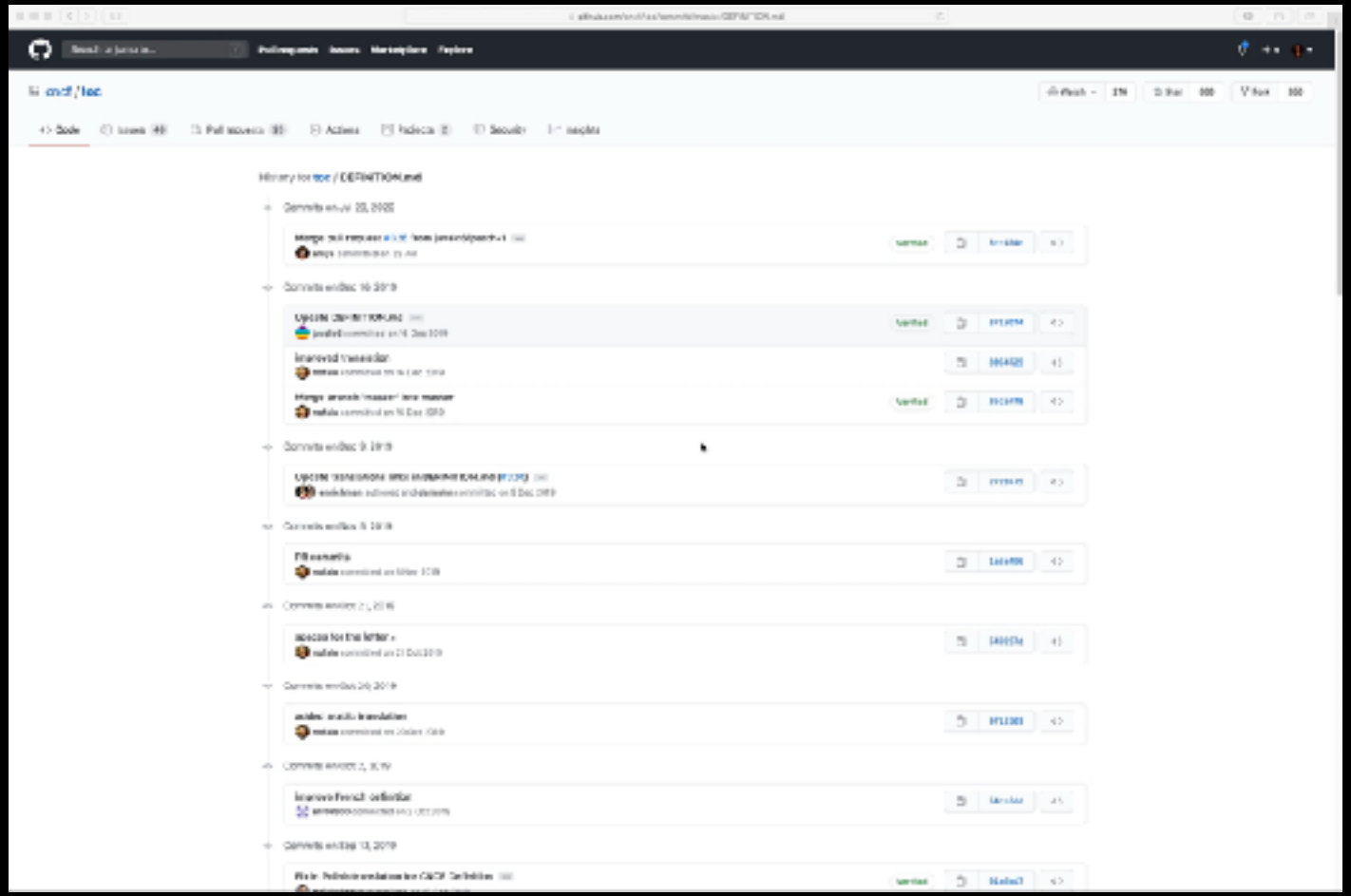
These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil.

The Cloud Native Computing Foundation seeks to drive adoption of this paradigm by fostering and sustaining an ecosystem of open source, vendor-neutral projects. We democratize state-of-the-art patterns to make these innovations accessible to everyone.

العربية

[illegible]



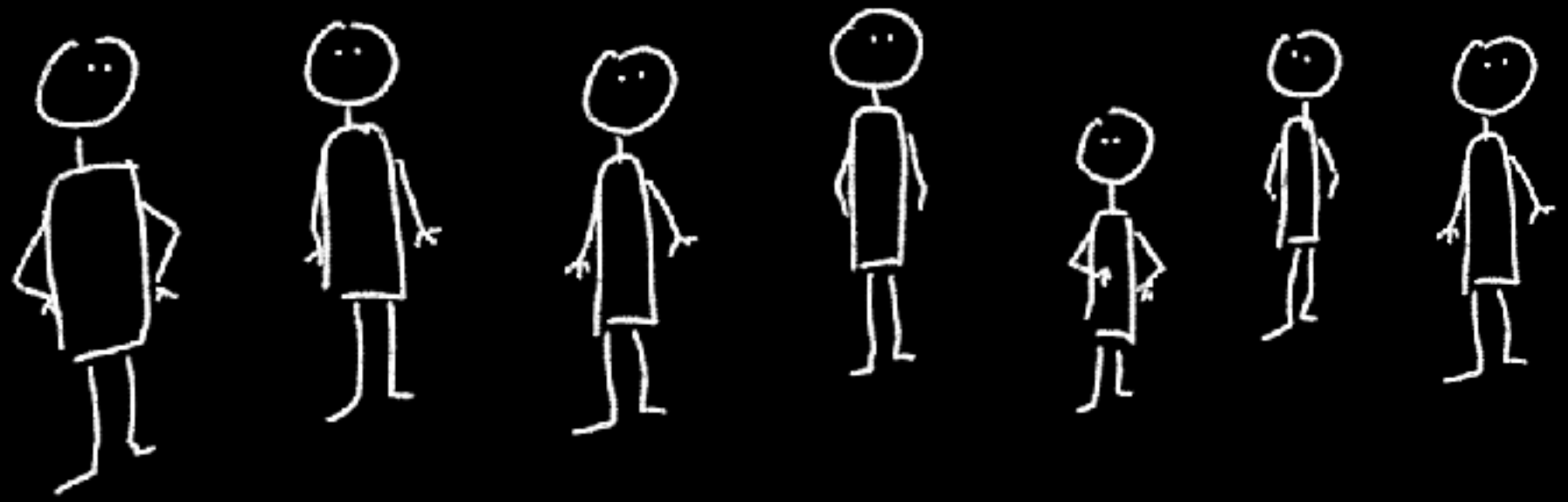


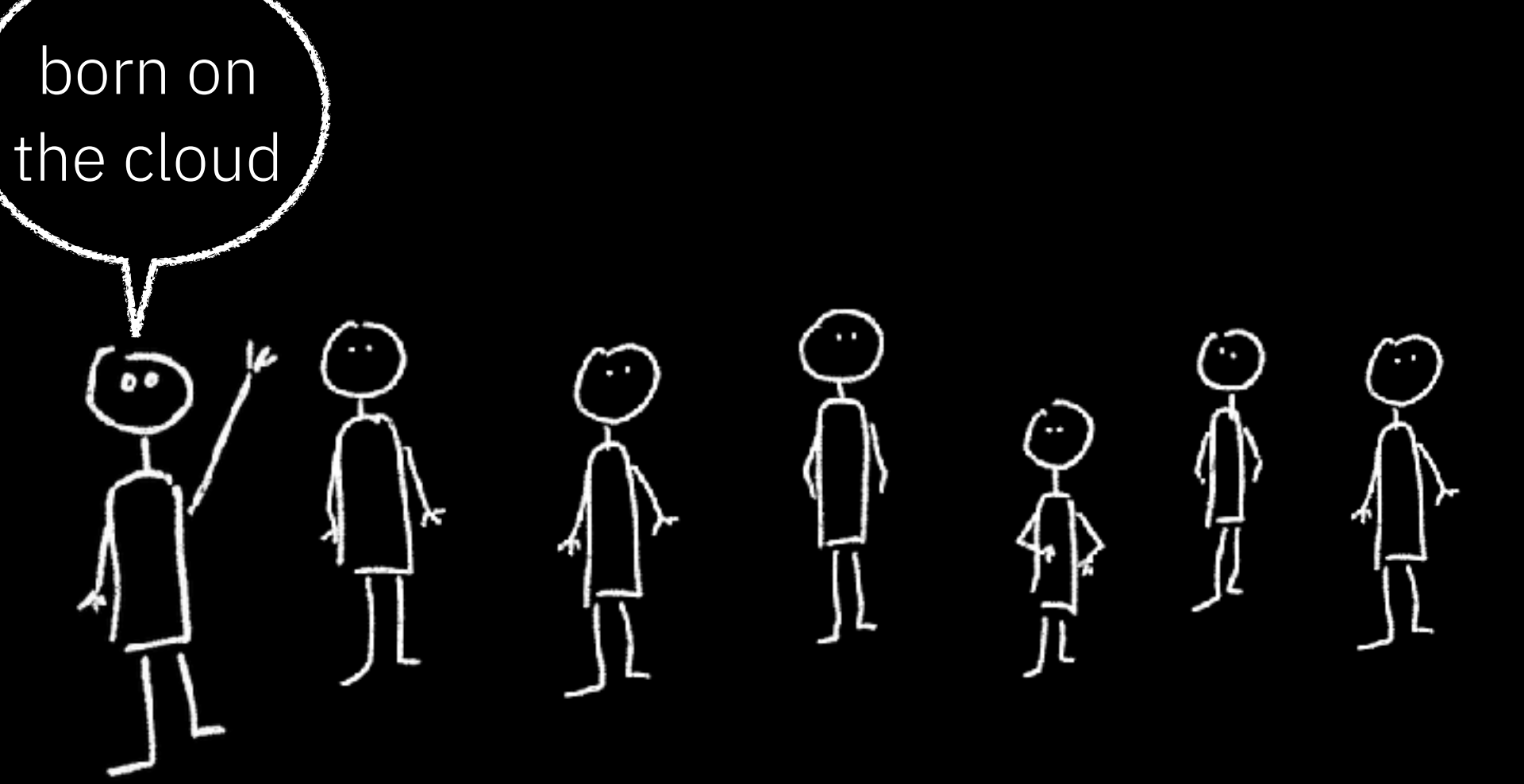
✓ **Create Cloud Native Definition**

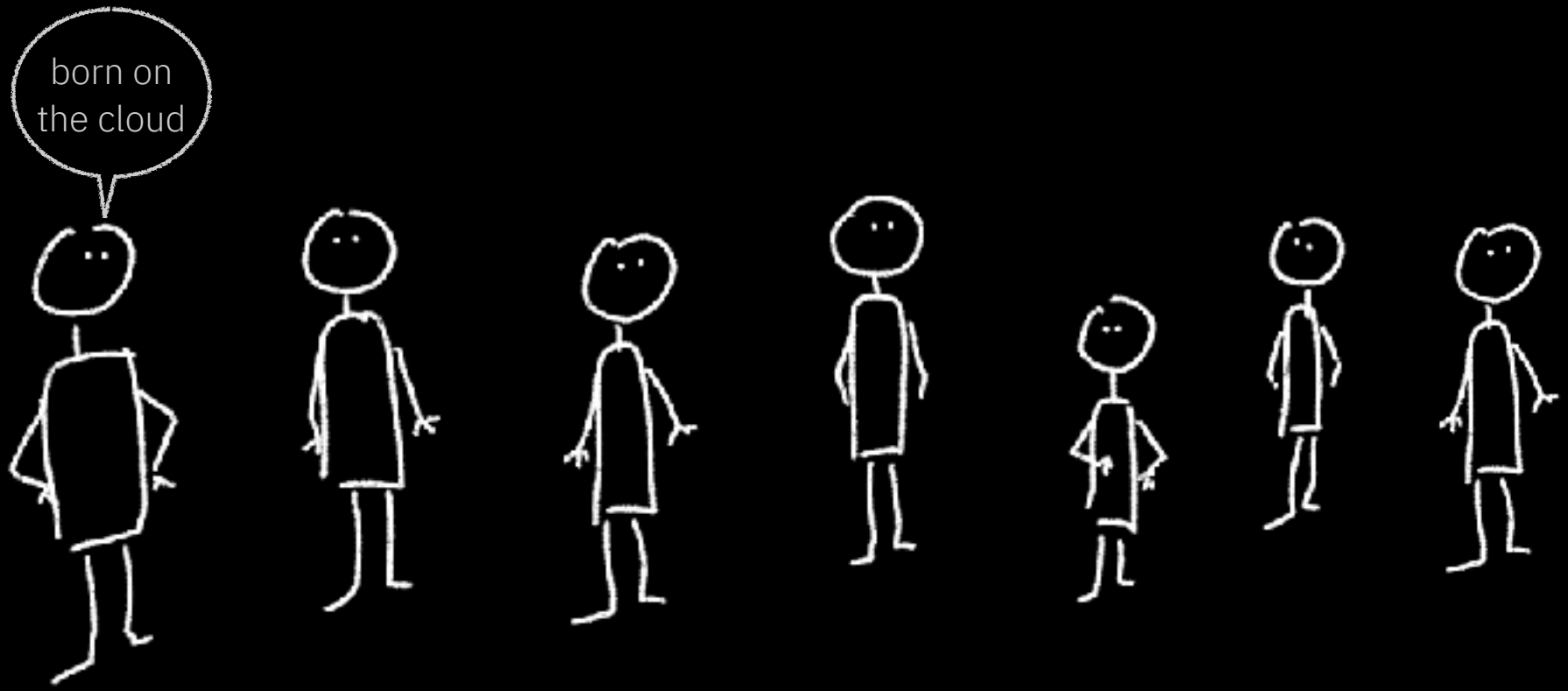
Based on 11 drafts from https://docs.google.com/document/d/1d9Ks3UvUV8sZj4rLbAMwmq0HZwi1CwnOZWGtrCuf0uk/

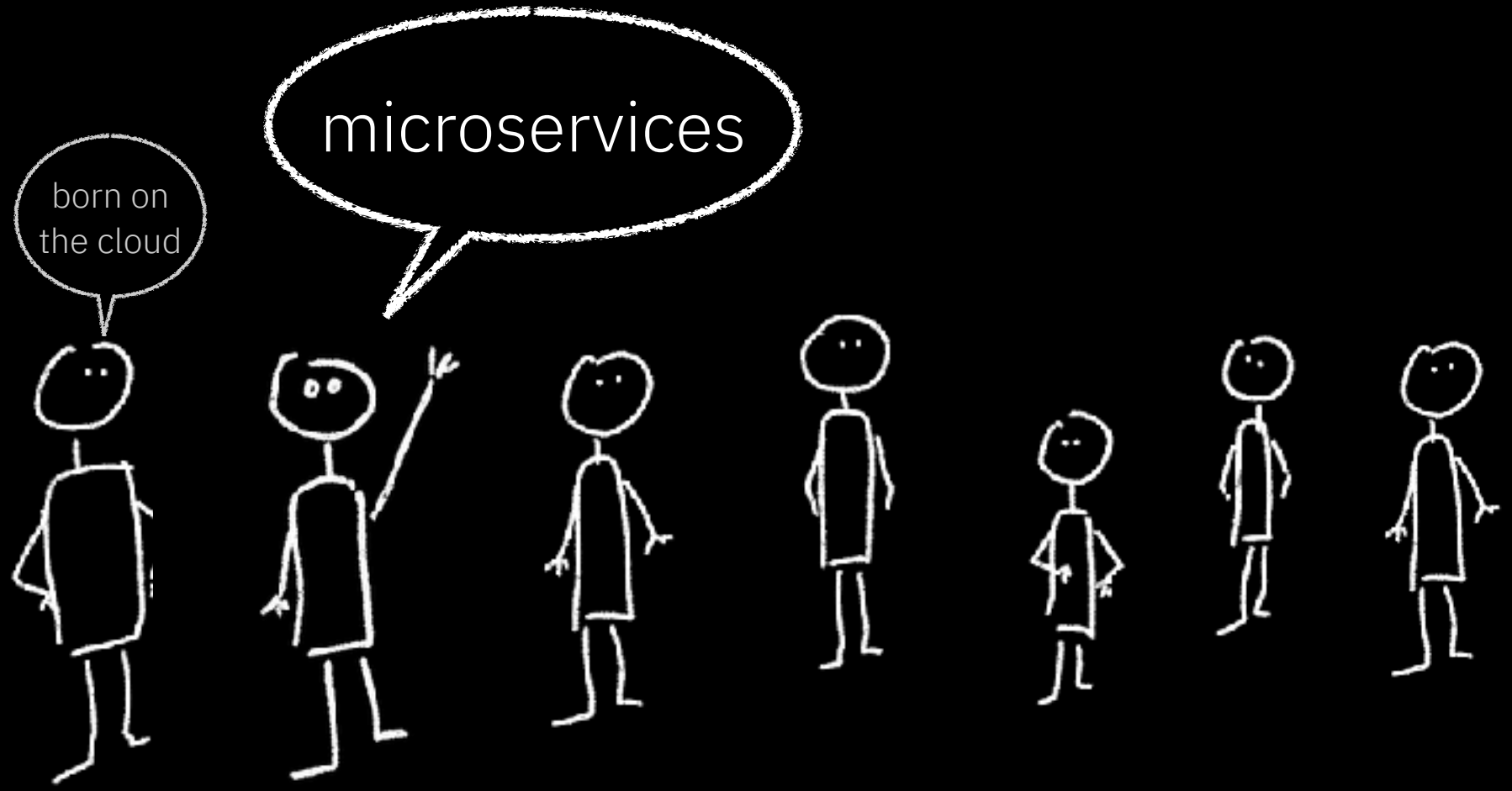
master (#117)

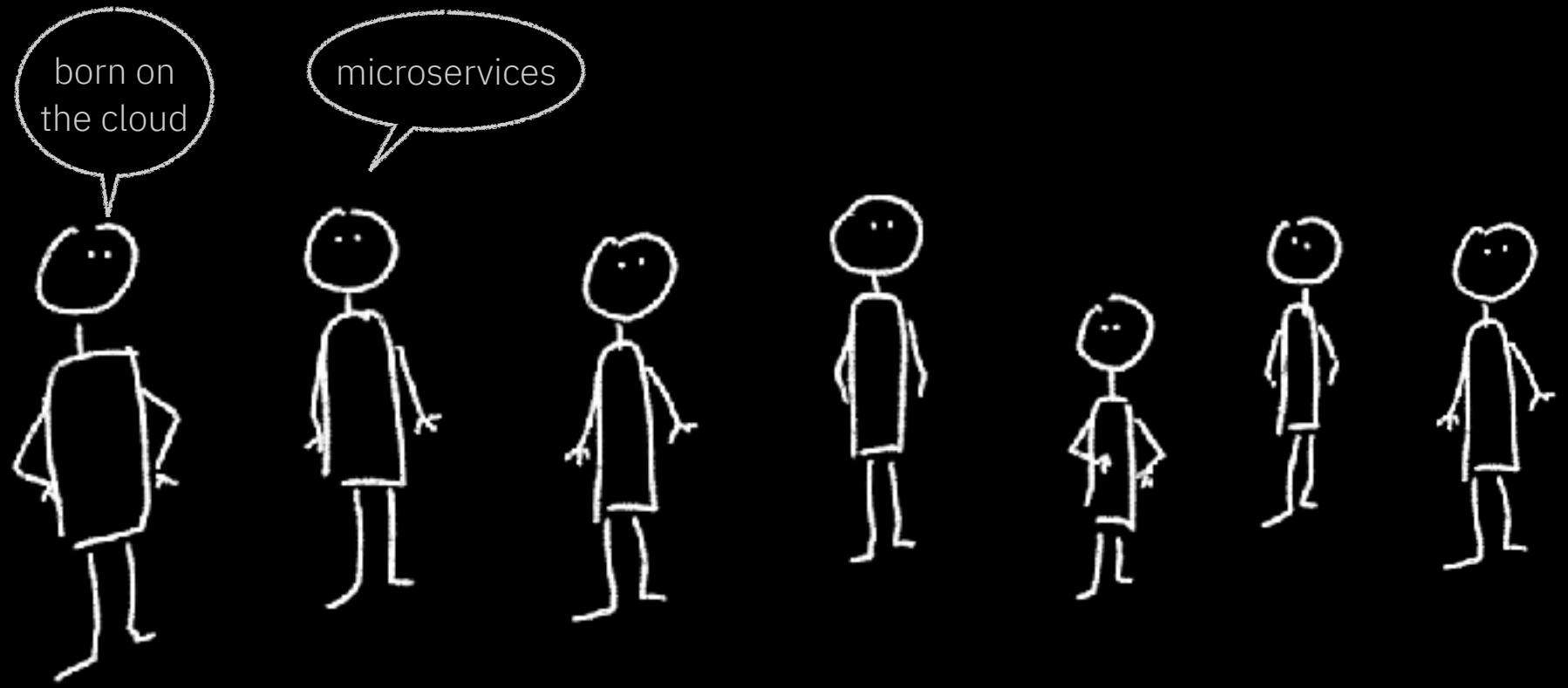
dankohn committed on 20 May 2018 Verified

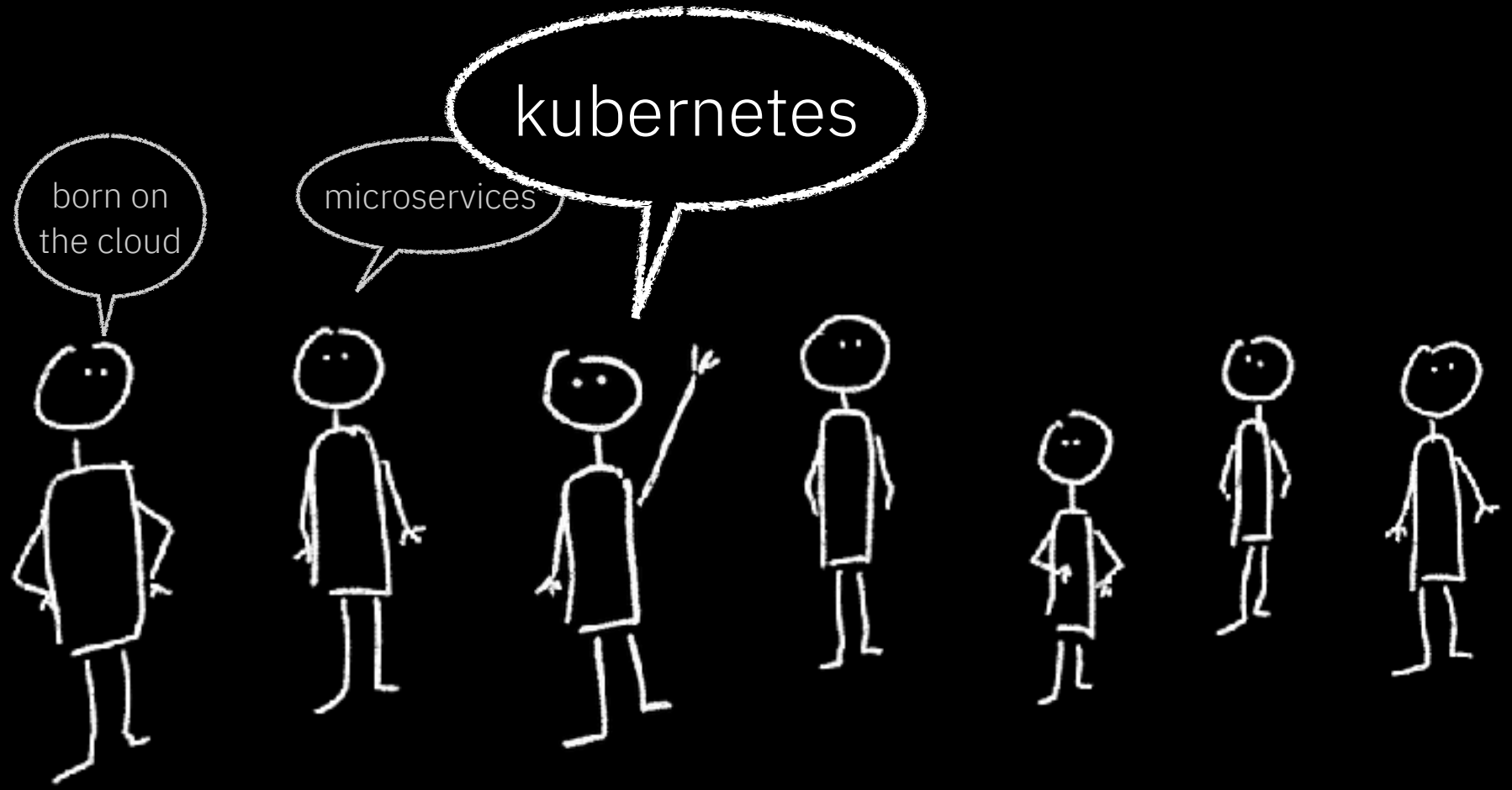


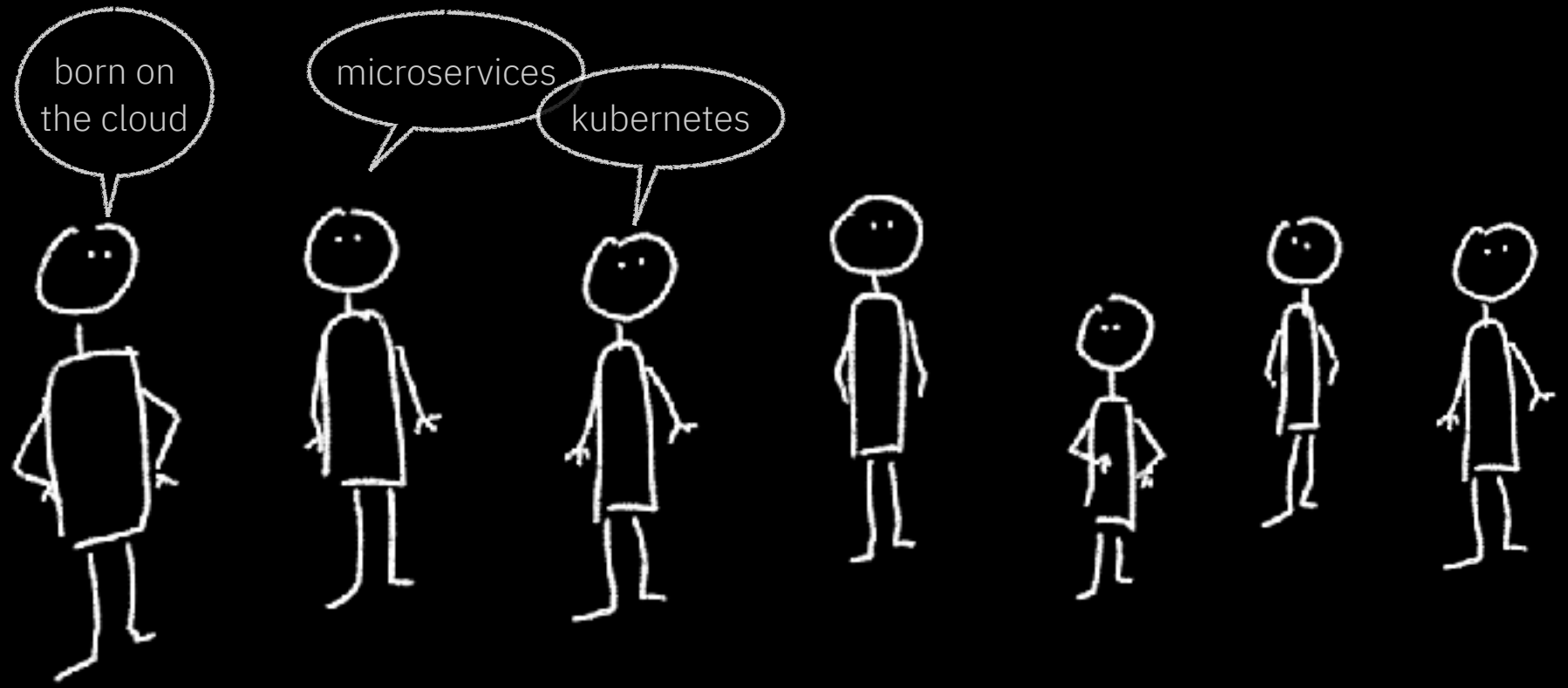

born on
the cloud

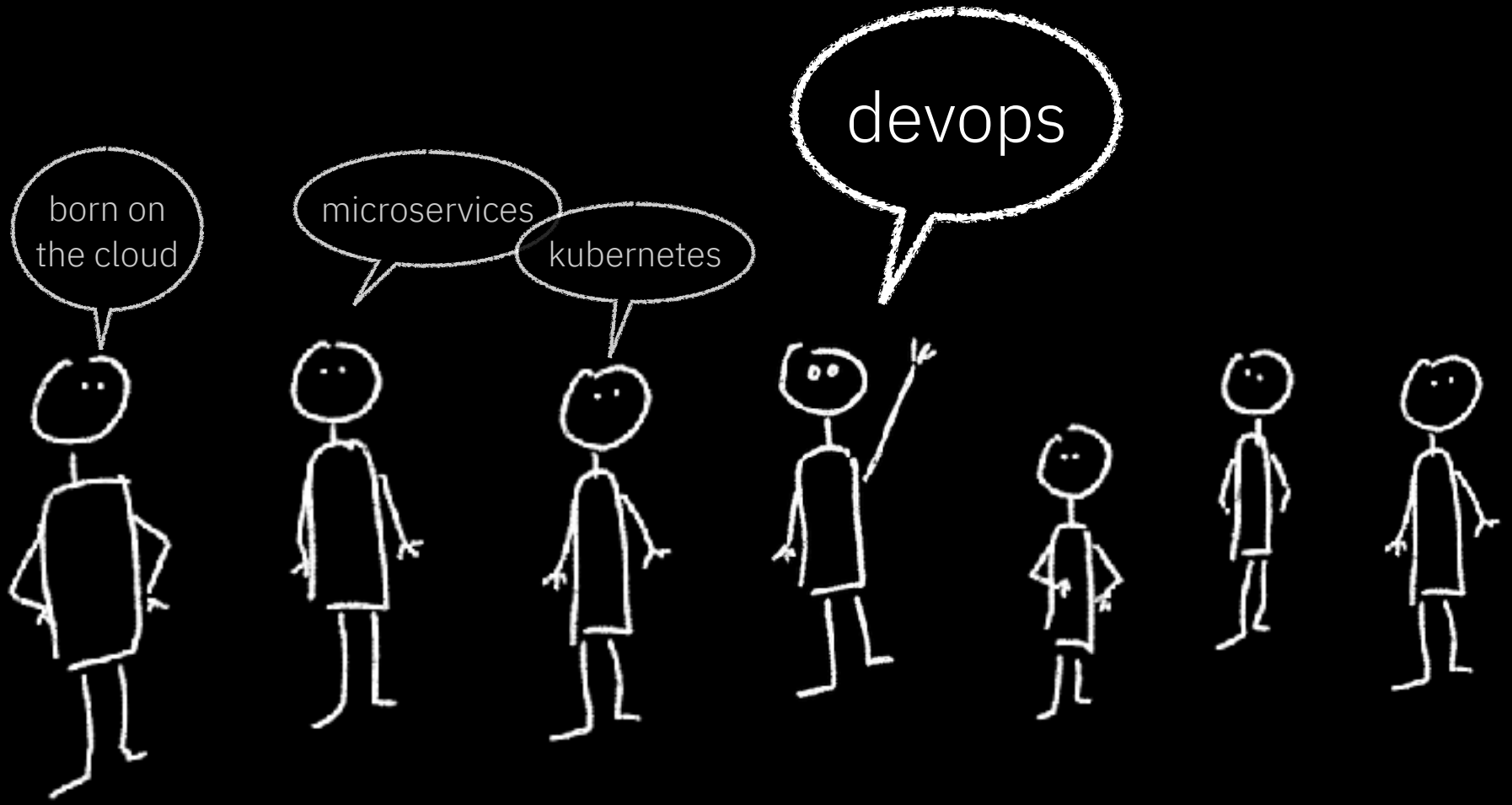

born on
the cloud

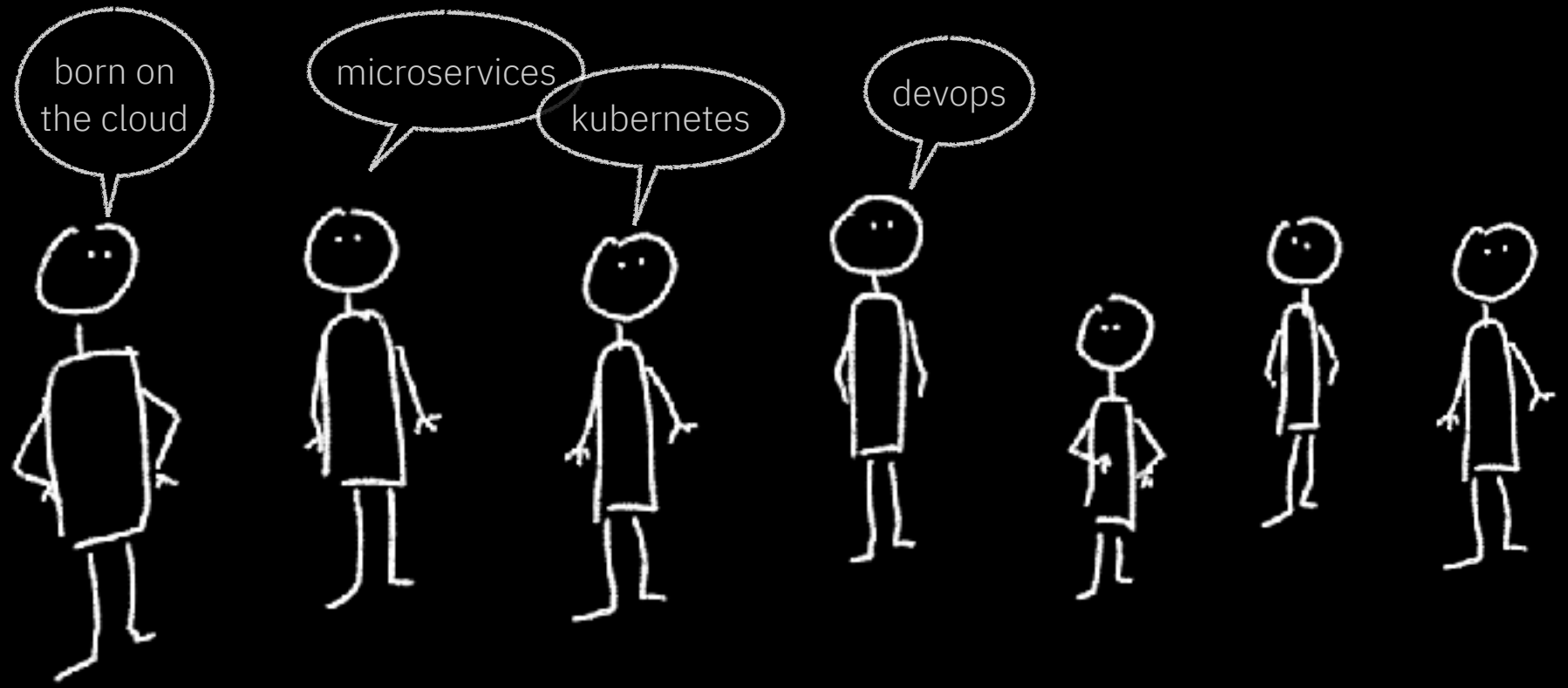

born on the cloud
microservices

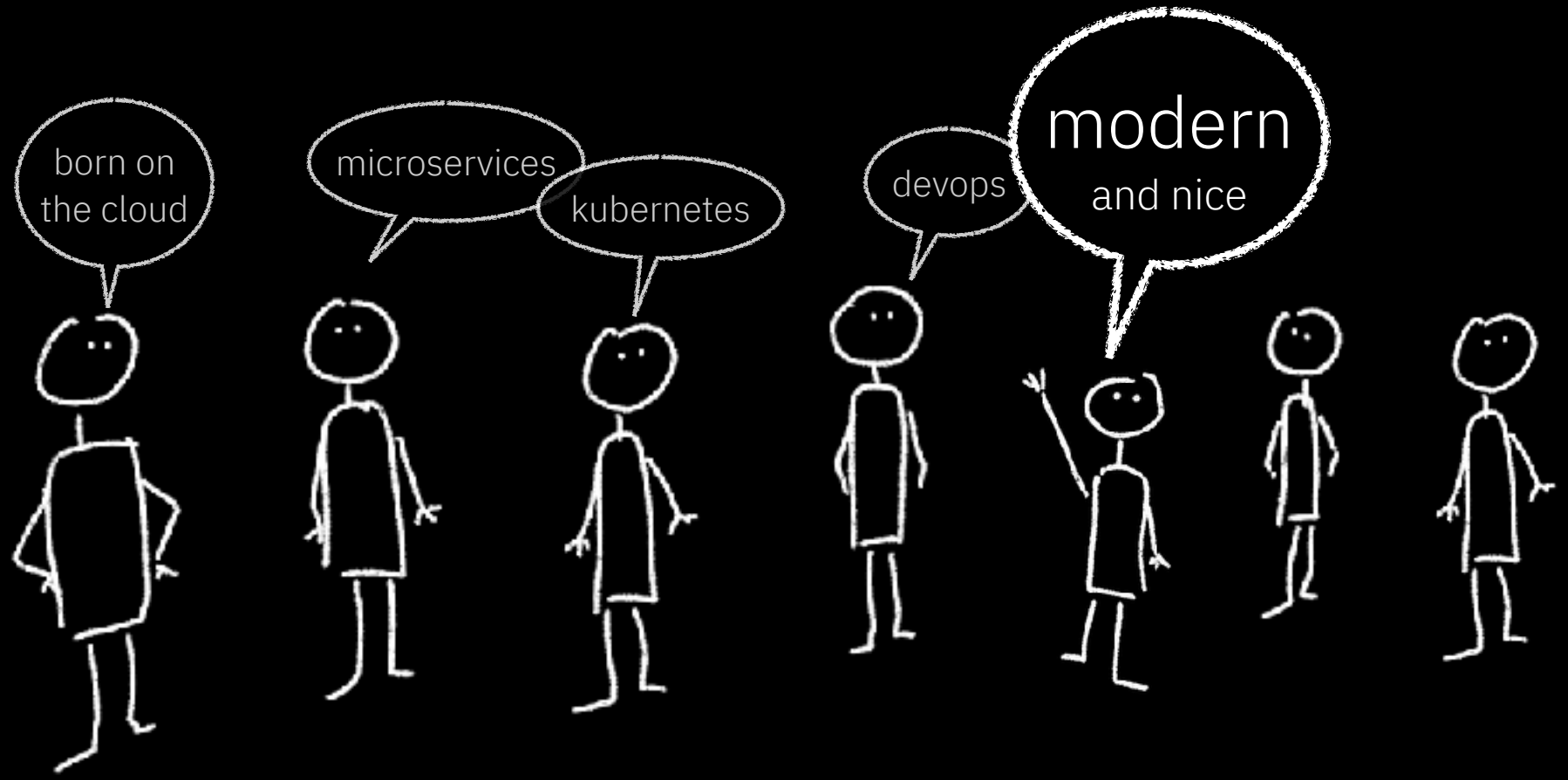

born on the cloud
microservices

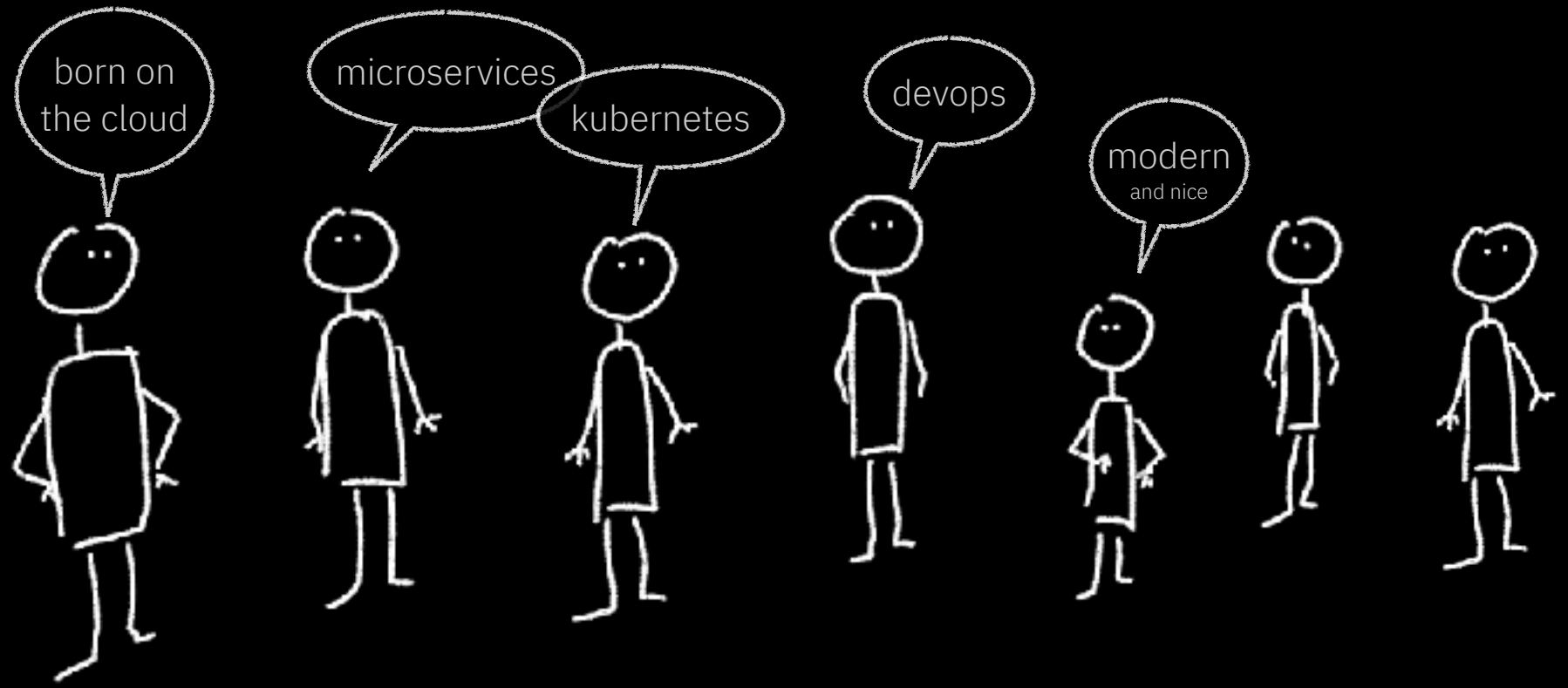

born on the cloud
microservices
kubernetes

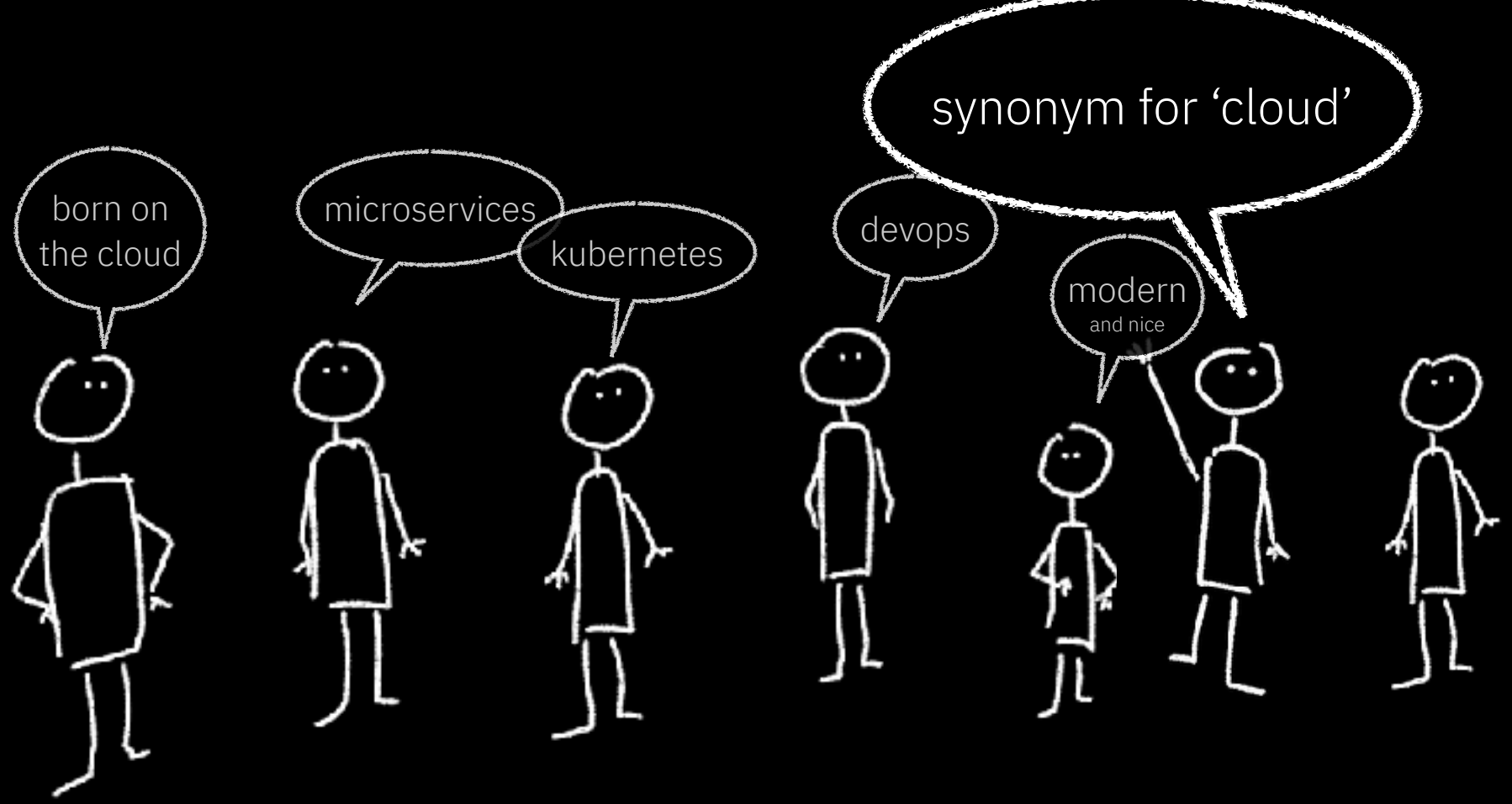

born on the cloud
microservices
kubernetes

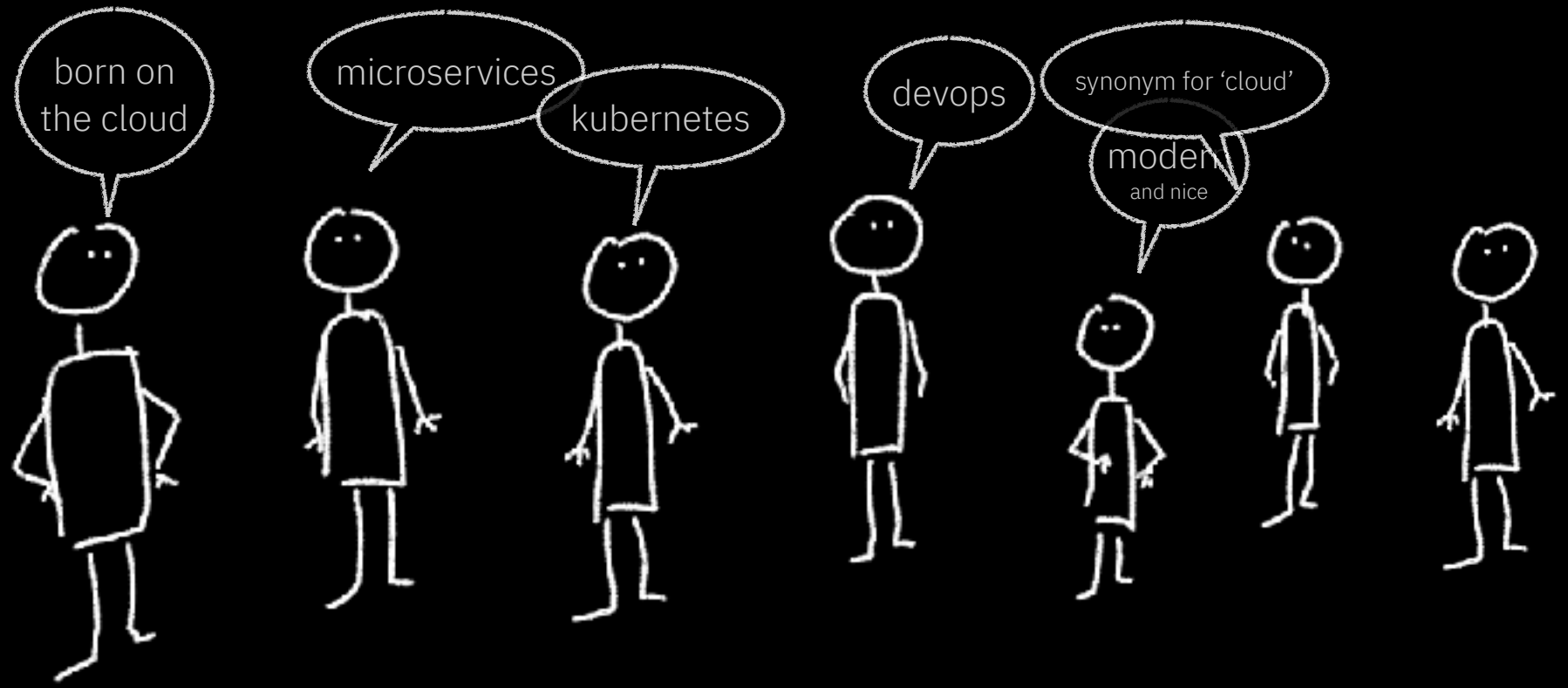

born on the cloud
microservices
kubernetes
devops

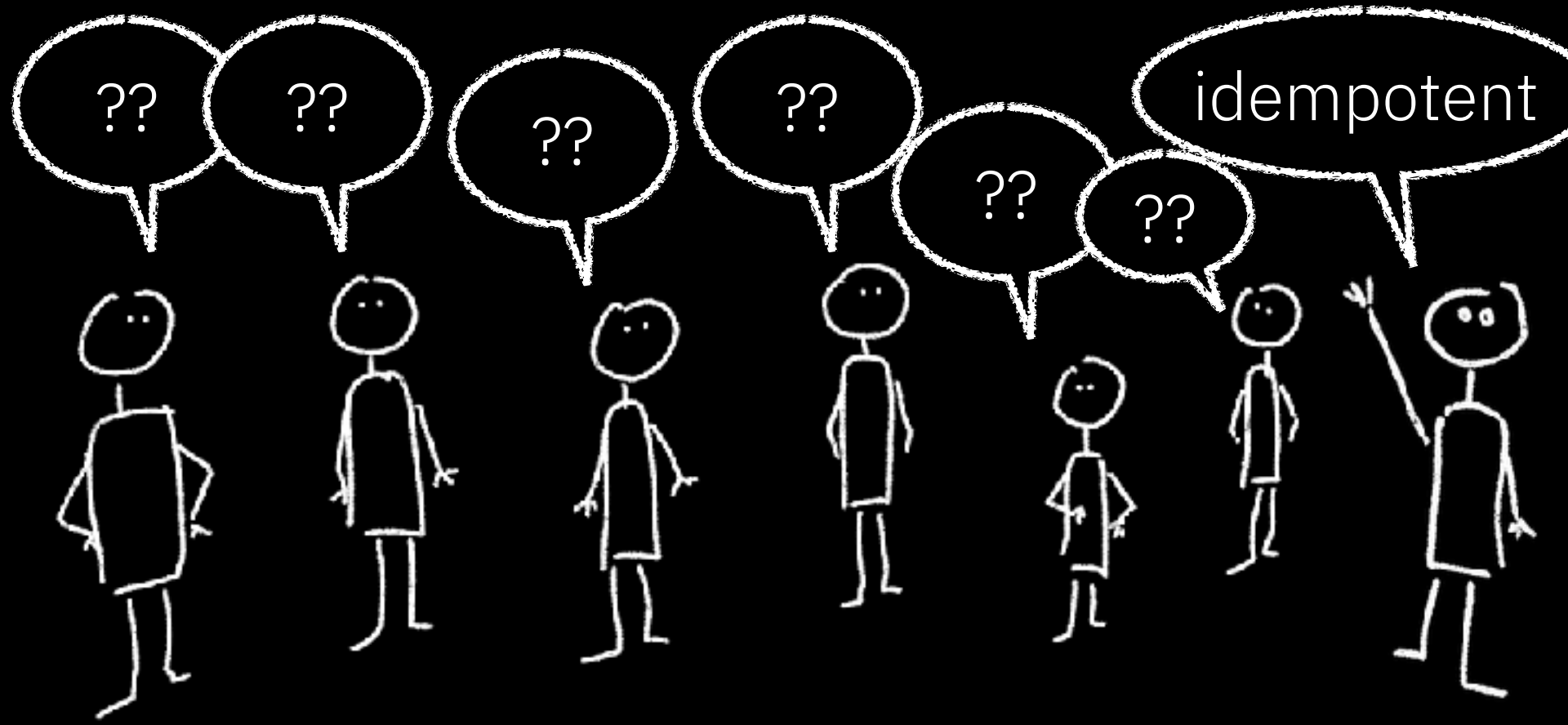

born on the cloud
microservices
kubernetes
devops

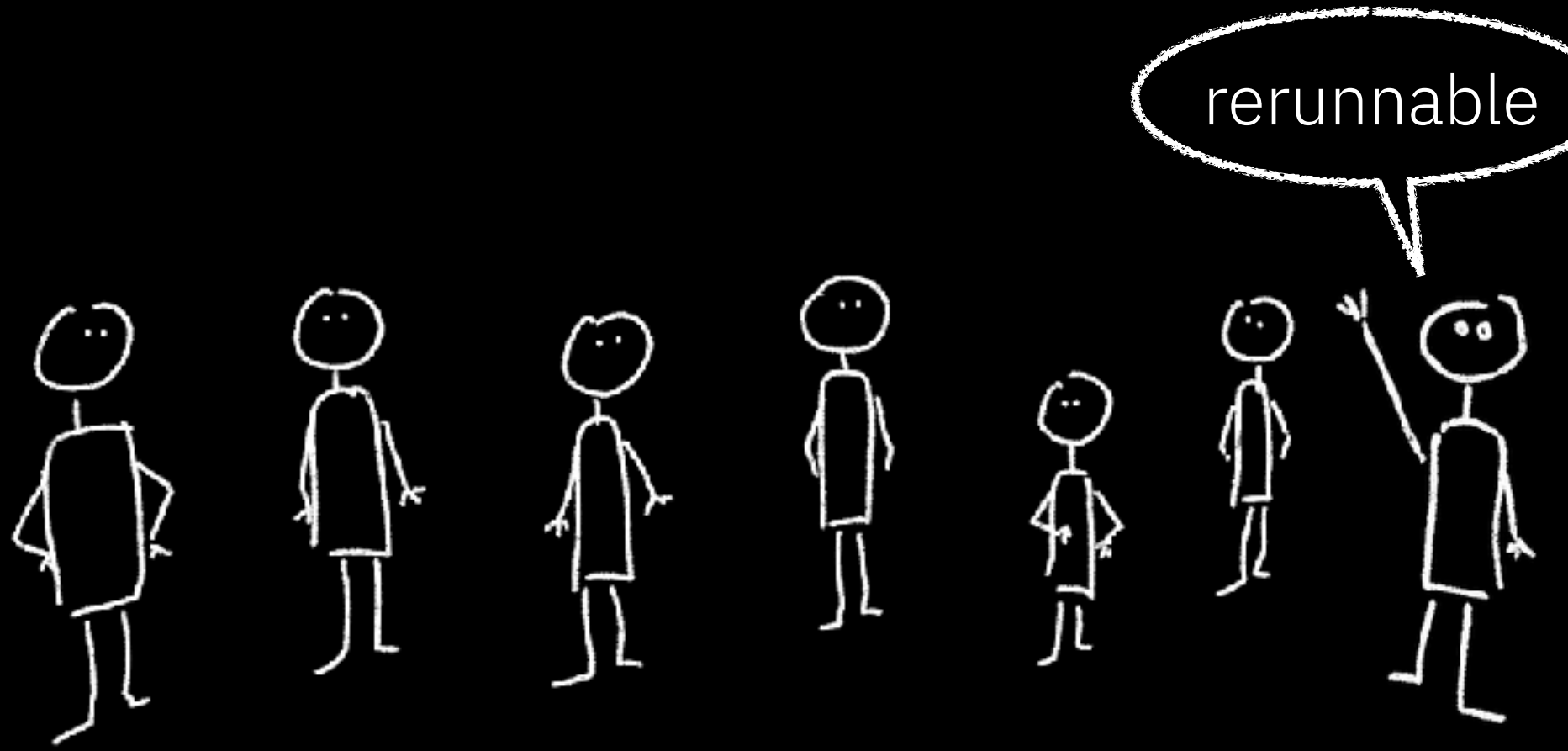

born on the cloud
microservices
kubernetes
devops
modern
and nice

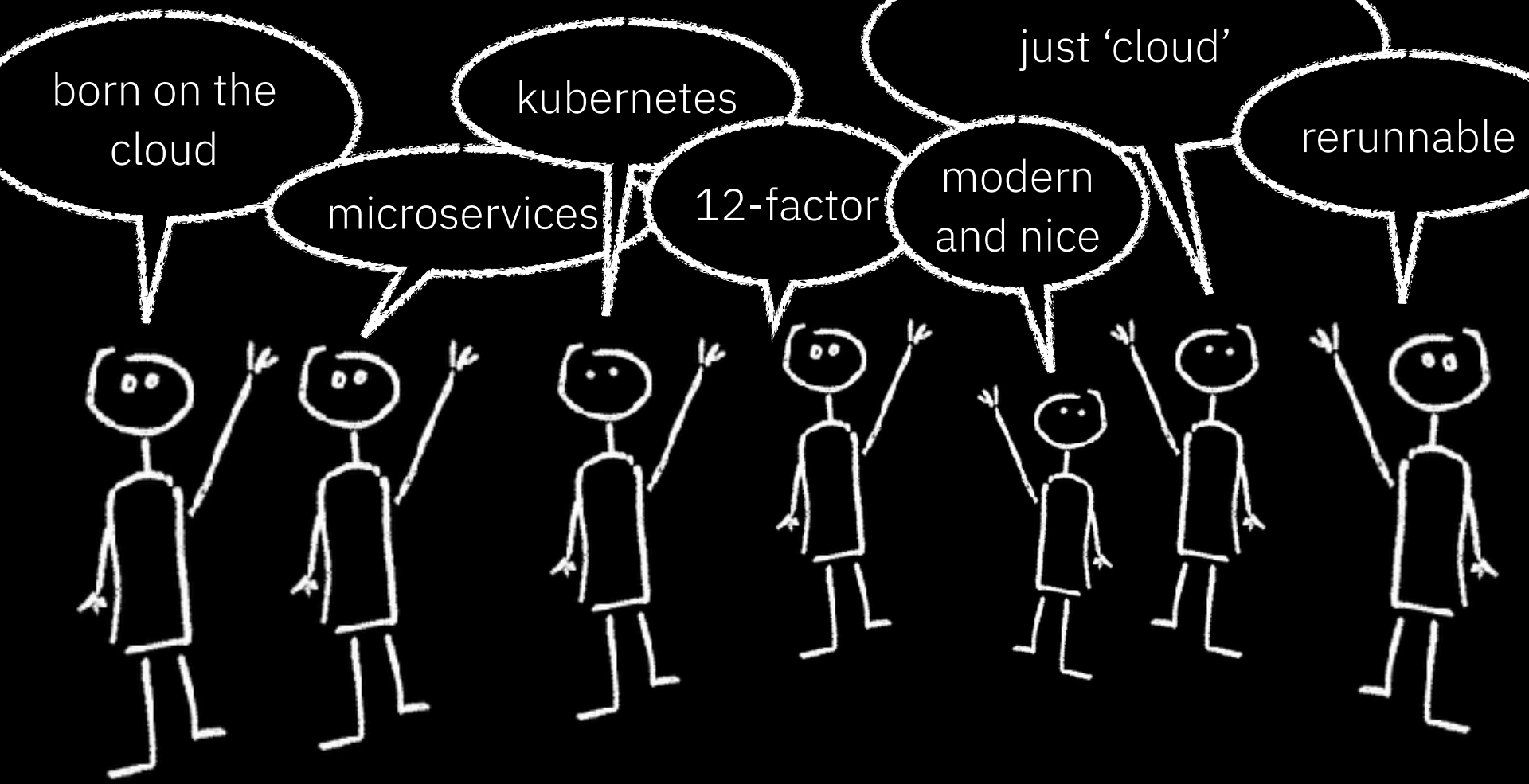

born on the cloud
microservices
kubernetes
devops
modern
and nice


synonym for ‘cloud’
born on the cloud
microservices
kubernetes
devops
modern
and nice


born on the cloud
microservices
kubernetes
devops
synonym for ‘cloud’
modern
and nice


??
??
??
??
??
??
idempotent


rerunnable


born on the cloud
microservices
kubernetes
12-factor
modern and nice
just ‘cloud’
rerunnable

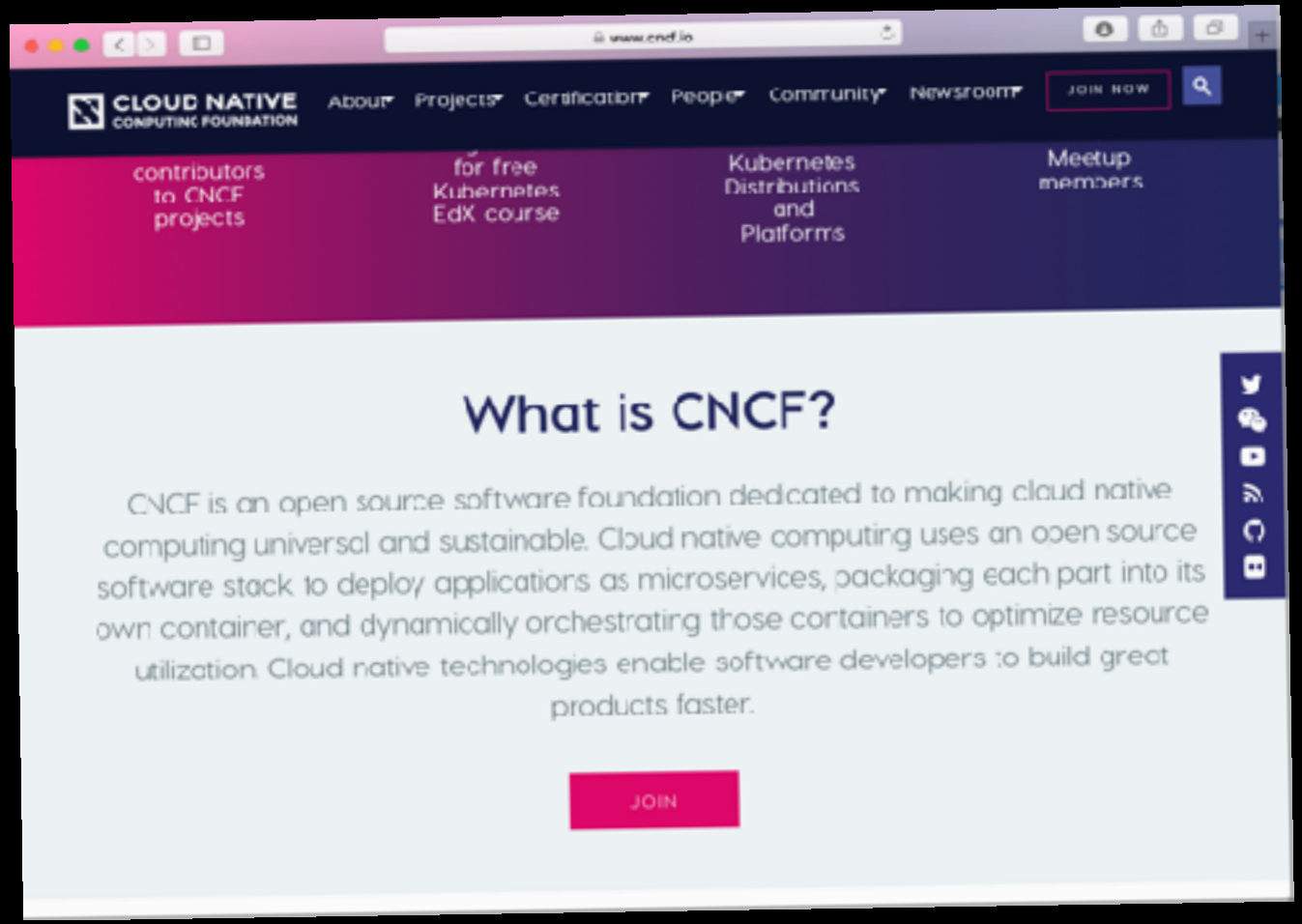
cloud native is **not**

cloud native is **not**
a synonym for ‘microservices’

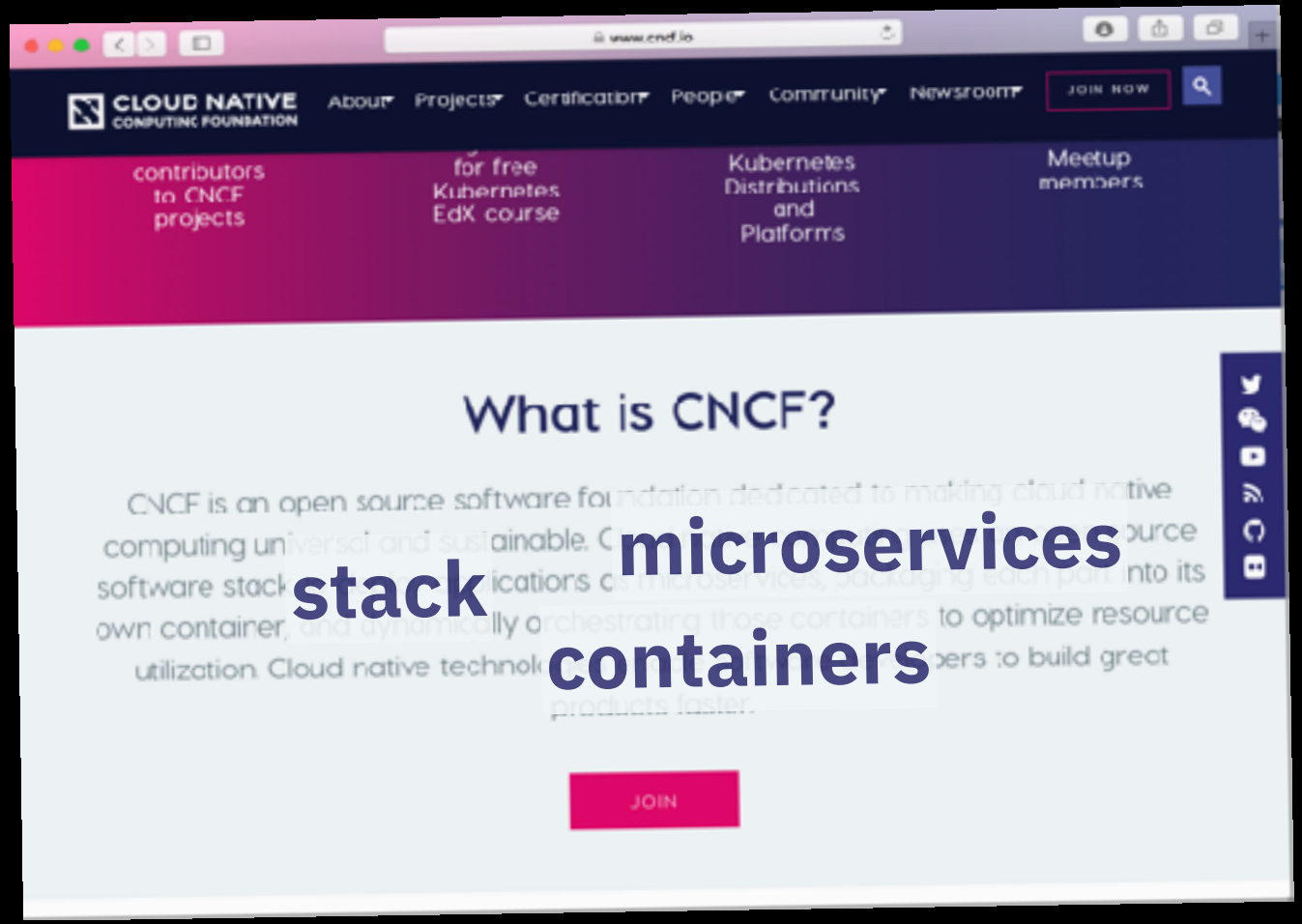
if ‘cloud native’ has to be a synonym for anything,
it would be ‘idempotent’

if ‘cloud native’ has to be a synonym for anything,
it would be ‘idempotent’

which definitely needs a synonym


www.cncf.io
CLOUD NATIVE
COMPUTING FOUNDATION
About
Projects
Certification
People
Community
Newsroom
JOIN NOW
contributors to CNCF projects
for free Kubernetes EdX course
Kubernetes Distributions and Platforms
Meetup members
What is CNCF?
CNCF is an open source software foundation dedicated to making cloud native computing universal and sustainable. Cloud native computing uses an open source software stack to deploy applications as microservices, packaging each part into its own container, and dynamically orchestrating those containers to optimize resource utilization. Cloud native technologies enable software developers to build great products faster.
JOIN

2019


www.cncf.io
CLOUD NATIVE
COMPUTING FOUNDATION
About
Projects
Certification
People
Community
Newsroom
JOIN NOW
contributors to CNCF projects
for free Kubernetes EdX course
Kubernetes Distributions and Platforms
Meetup members
What is CNCF?
stack
microservices
containers
JOIN

2019

112 lines (68 sloc) 12.4 KB

Raw Blame

CNCF Cloud Native Definition v1.0

Approved by TOC: 2018-06-11

العربية (Arabic) | 中文版本 (Chinese) | 日本語版 (Japanese) | 한국어 (Korean) | Deutsch (German) | Español (Spanish) Français (French) | Polski (Polish) | Português Brasileiro (Portuguese) | Русский (Russian)

Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.

These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil.

2020

112 lines (68 sloc) 12.4 KB

Raw Blame

CNCF Cloud Native Definition v1.0

Approved by TOC: 2018-06-11

العربية (Arabic) | 中文版本 (Chinese) | 日本語版 (Japa

Français (French) | Polski (Polish) | Português Brasileiro (Portuguese) | Русский (Russian)

Cloud native technolo e applications in m odern, dynamic environments such as public, private, and hybrid clo uds. Containers, service meshes, microservices, immutable infrastruct his approach.

These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil.

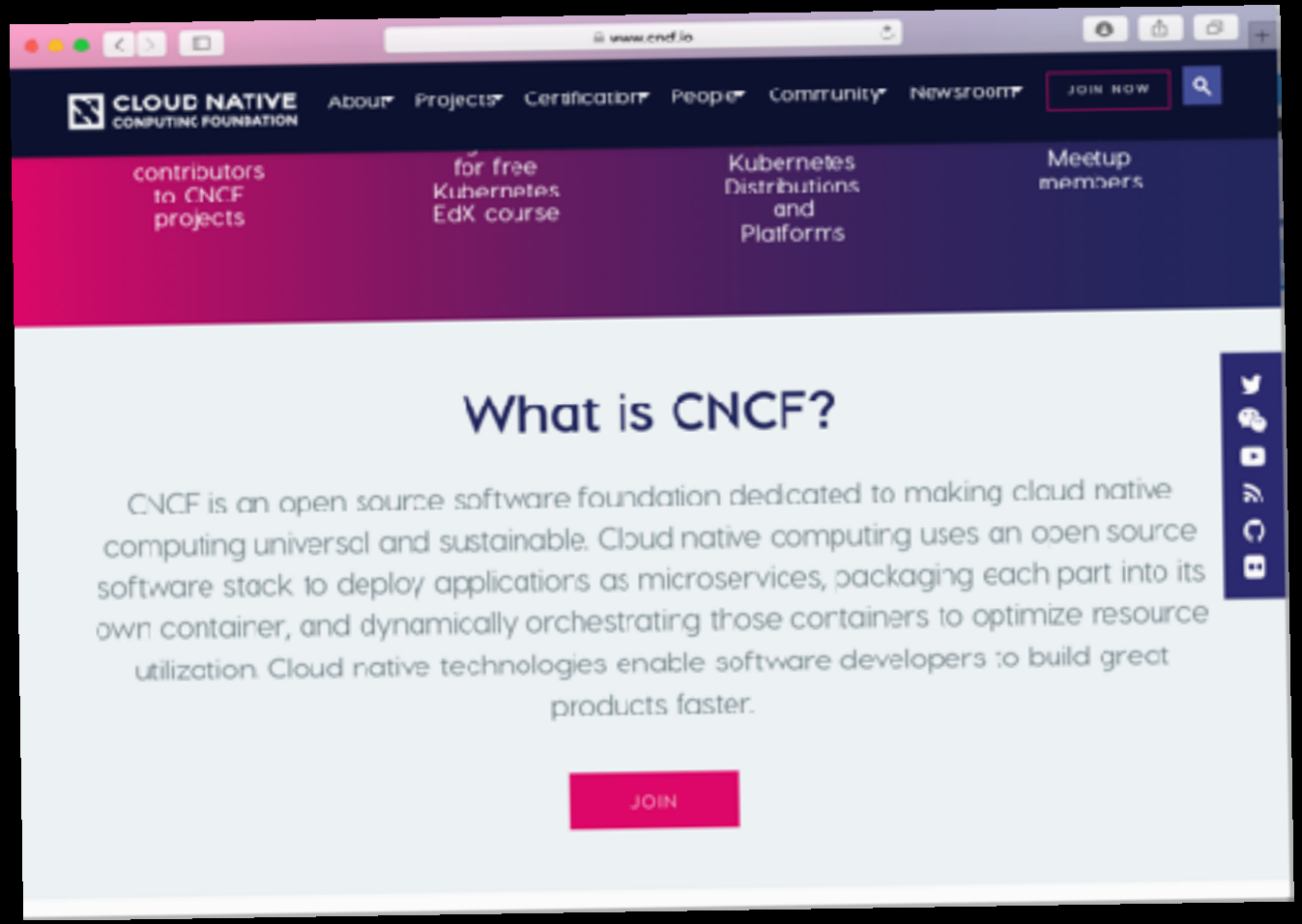
immutable infrastructure

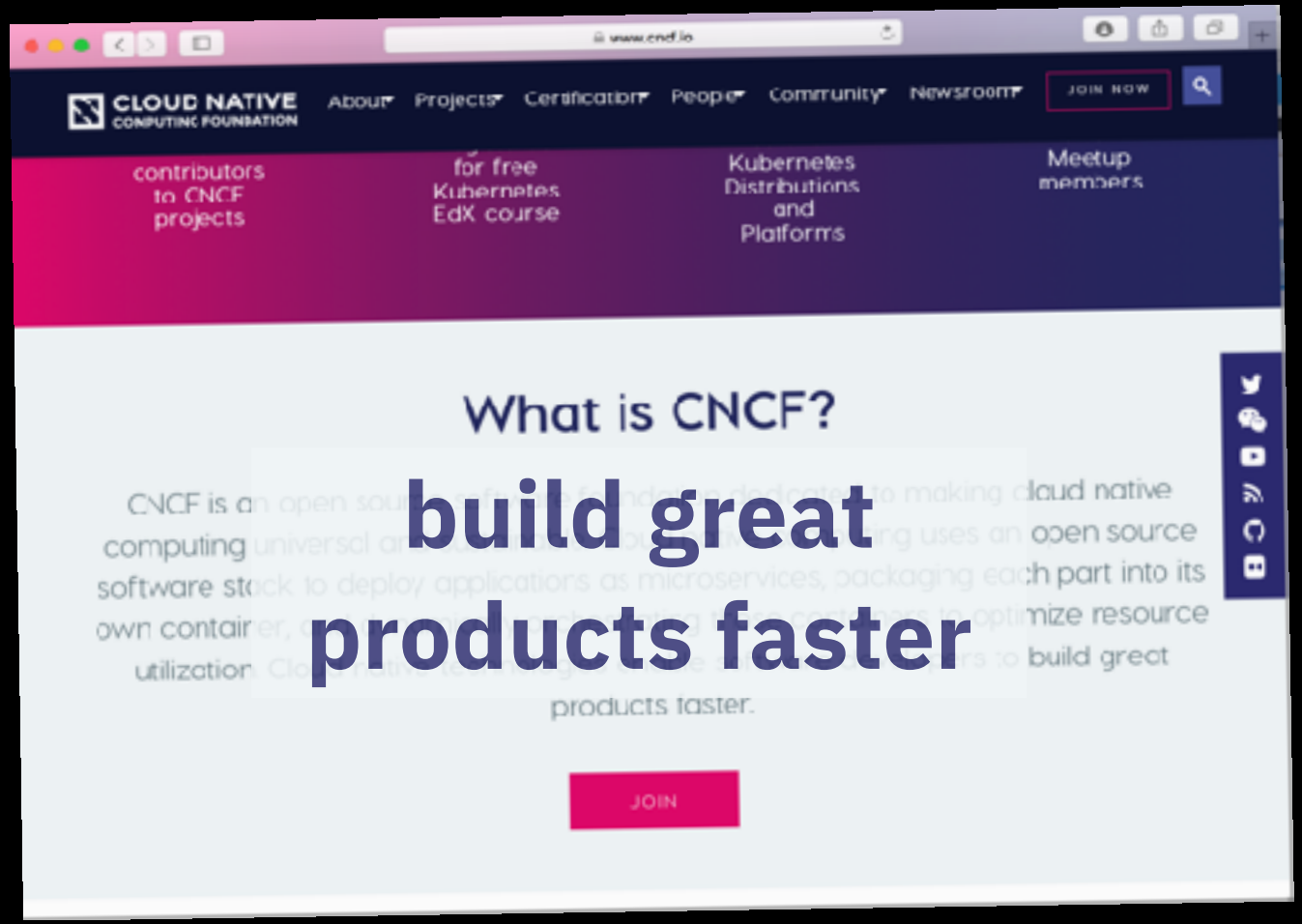
microservices

exemplify

2020

why?


www.cncf.io
CLOUD NATIVE
COMPUTING FOUNDATION
About
Projects
Certification
People
Community
Newsroom
JOIN NOW
contributors to CNCF projects
for free Kubernetes EdX course
Kubernetes Distributions and Platforms
Meetup members
What is CNCF?
CNCF is an open source software foundation dedicated to making cloud native computing universal and sustainable. Cloud native computing uses an open source software stack to deploy applications as microservices, packaging each part into its own container, and dynamically orchestrating those containers to optimize resource utilization. Cloud native technologies enable software developers to build great products faster.
JOIN



build great products faster

112 lines (68 sloc) 12.4 KB

Raw Blame

CNCF Cloud Native Definition v1.0

Approved by TOC: 2018-06-11

العربية (Arabic) | 中文版本 (Chinese) | 日本語版 (Japanese) | 한국어 (Korean) | Deutsch (German) | Español (Spanish) Français (French) | Polski (Polish) | Português Brasileiro (Portuguese) | Русский (Russian)

Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.

These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil.

2020

112 lines (68 sloc) 12.4 KB

Raw Blame

CNCF Cloud Native Definition v1.0

Approved by TOC: 2018-06-11

العربية (Arabic) | 中文 | 日本語 | 한국어 | Deutsch | Ελληνικά | Español | Français (French) | Polski (Polish) | Português Brasileiro (Portuguese) | Русский (Russian)

Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.

These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil.

make high-impact changes frequently and predictably with minimal toil

2020

IBM **Garage**

@holly_cummins

what **problem** are we trying to solve?

fail

the muddy goal

what **problem** are we trying to solve?

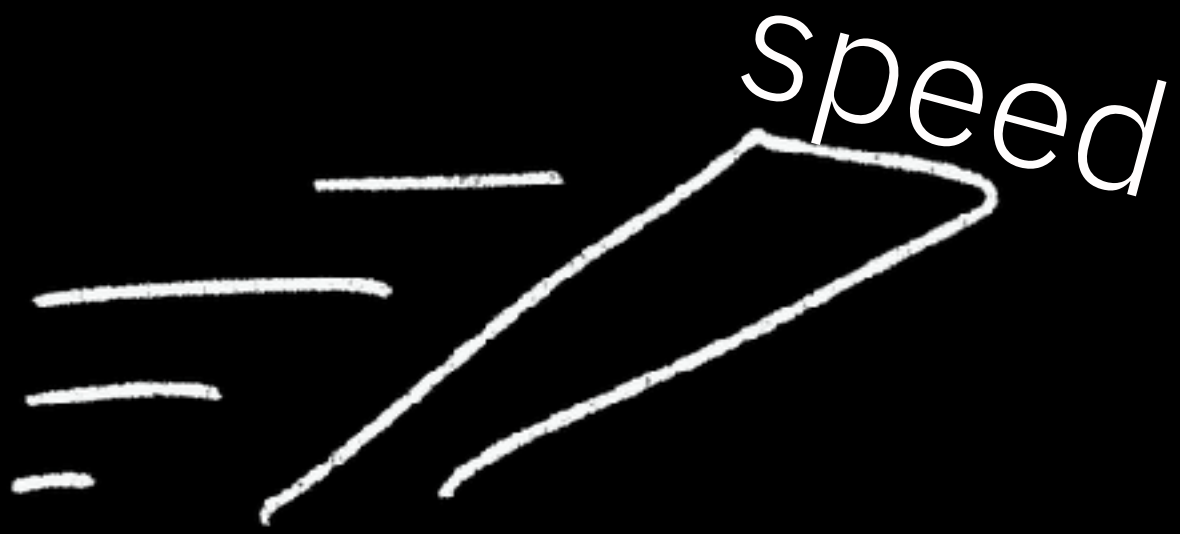
“everyone else is doing it?”

wishful mimicry

why cloud?

cost

e l a s t i c i t y


Speed

exotic capabilities

why cloud native?

2011

12

factors

2011

12 factors

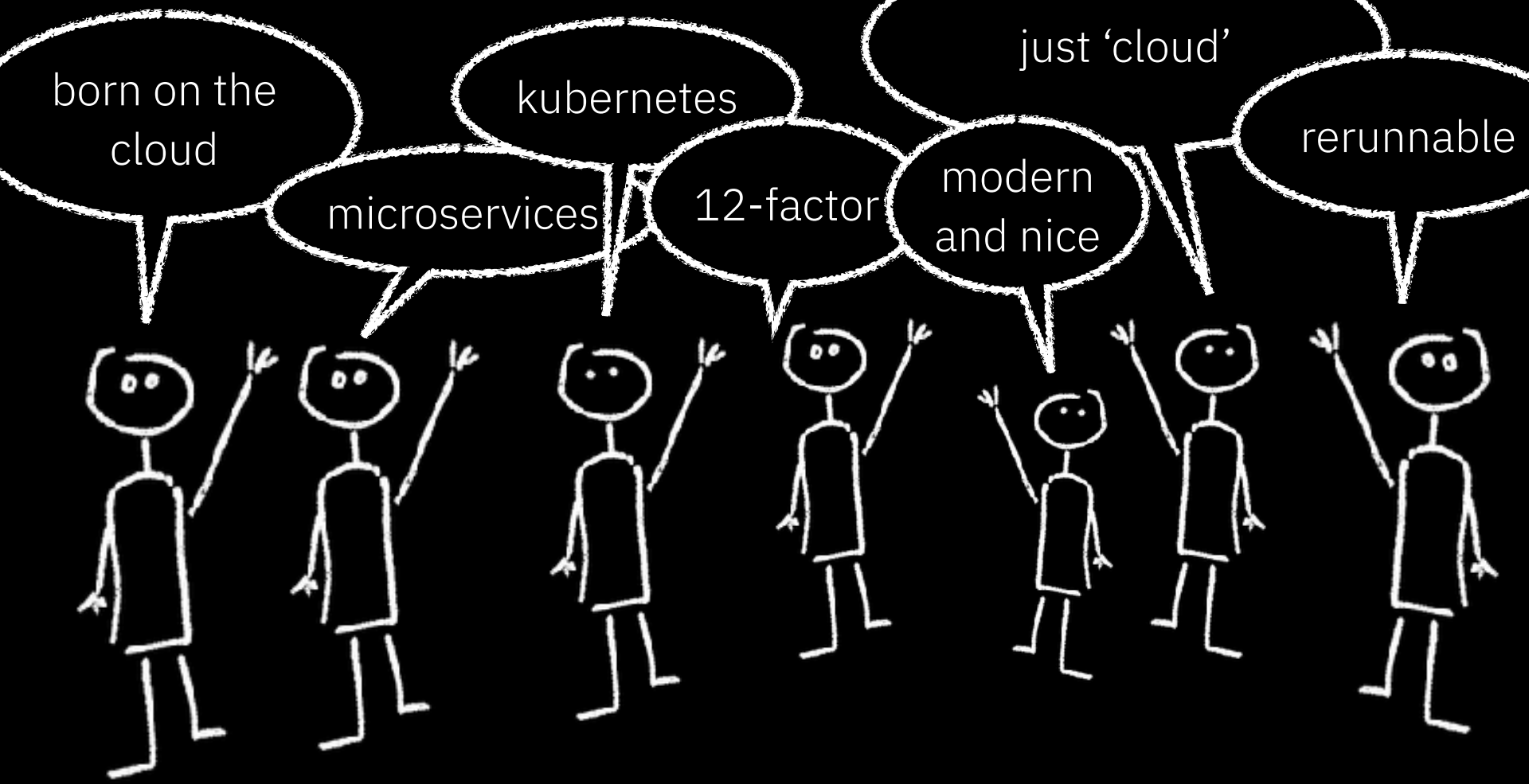
how to write a cloud application so you don’t get electrocuted

2010 the dawn of cloud native

Ian Cooper @ICooper · 1h

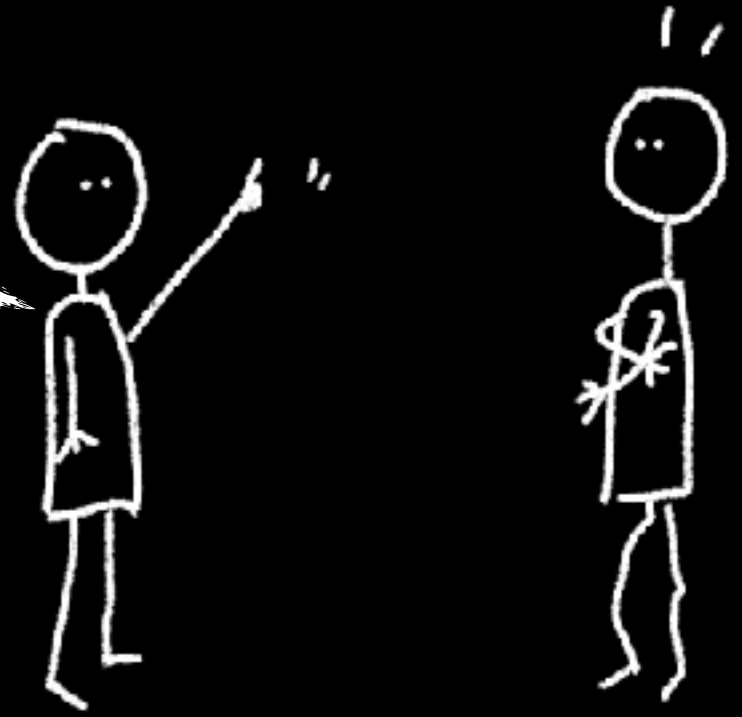
Replying to @samnewman

It does seem perverse that Cloud Native, which I thought origins just implied 'built to operate in cloud infrastructure' as opposed to say, 'lift-and-shift from a data centre', now means 'runs on Kubernetes.'.

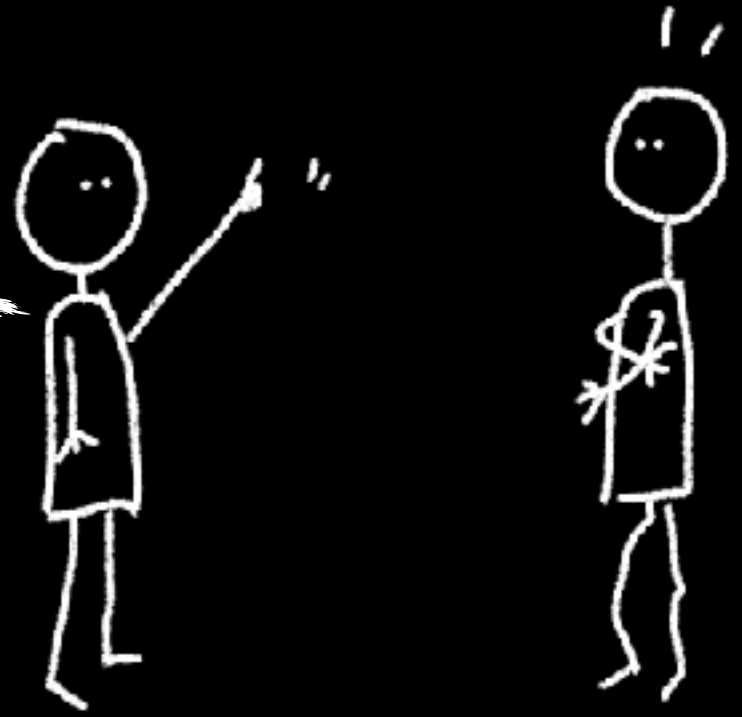

born on the cloud
microservices
kubernetes
12-factor
modern and nice
just ‘cloud’
rerunnable

are we all agreed on the goal?

why are there no **microservices** in this cloud native app Alice?



why is the cloud **only** saving us money, Alice?



fail

microservices envy

microservices are not the **goal**

microservices
are not the **goal**

they are the **means**

“we’re going too slowly.

we need to get rid of COBOL and make microservices!”

“we’re going too slowly.

we need to get rid of COBOL and make microservices!”

“… but our release board only meets twice a year.”

containers are a good base

it’s not a **competition** to see how many you can have

containers are a good base

distributed monolith

distributed monolith

but without compile-time checking
... or guaranteed function execution

reasons **not** to do microservices

small team

not planning to release independently

don’t want complexity of a service mesh - or worse yet, rolling your own

domain model doesn’t split nicely

fail

cloud-native spaghetti

“every time we change one microservice, another breaks”

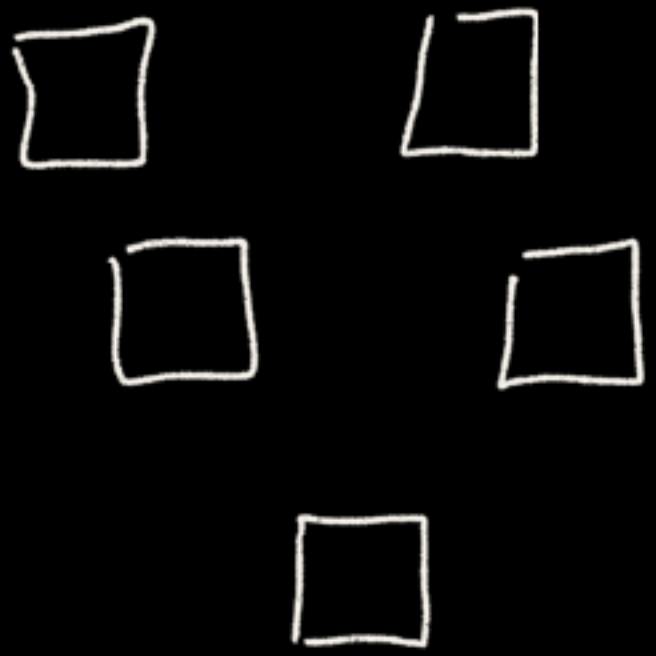
cloud-native spaghetti is still spaghetti

(Image: Cloudy with a Chance of Meatballs.)

distributed != decoupled

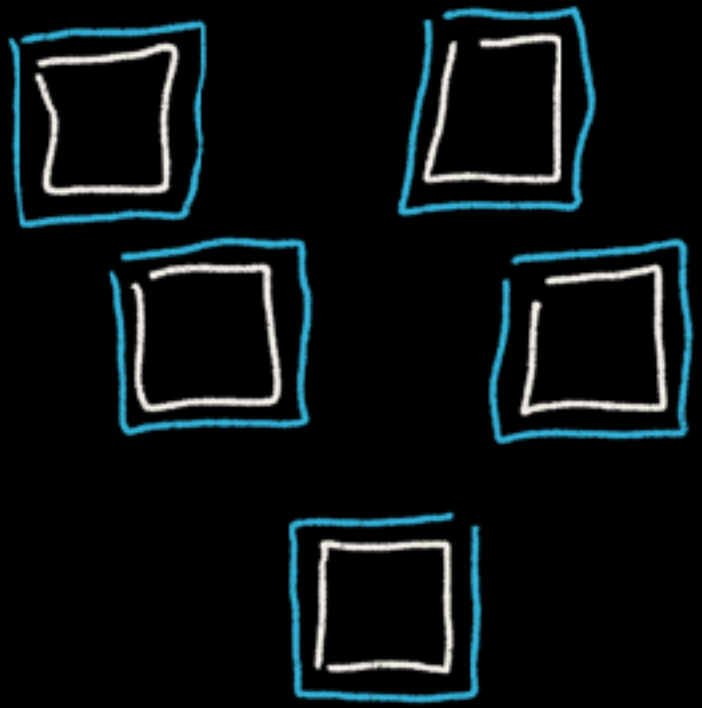
“each of our microservices has duplicated the same object model … with twenty classes and seventy fields”

Microservice



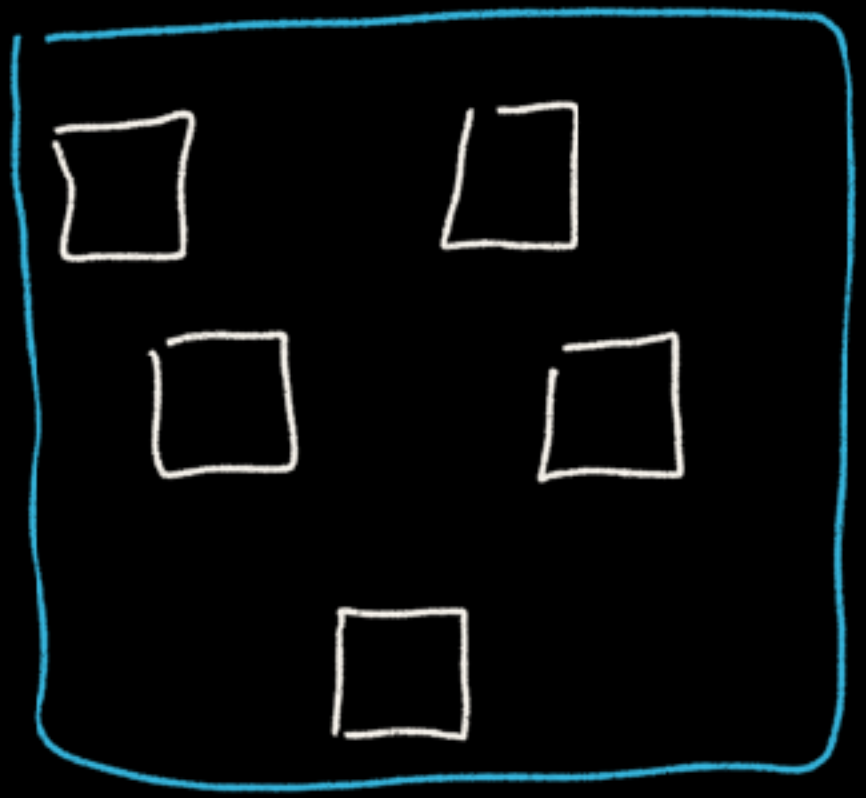
Microservice

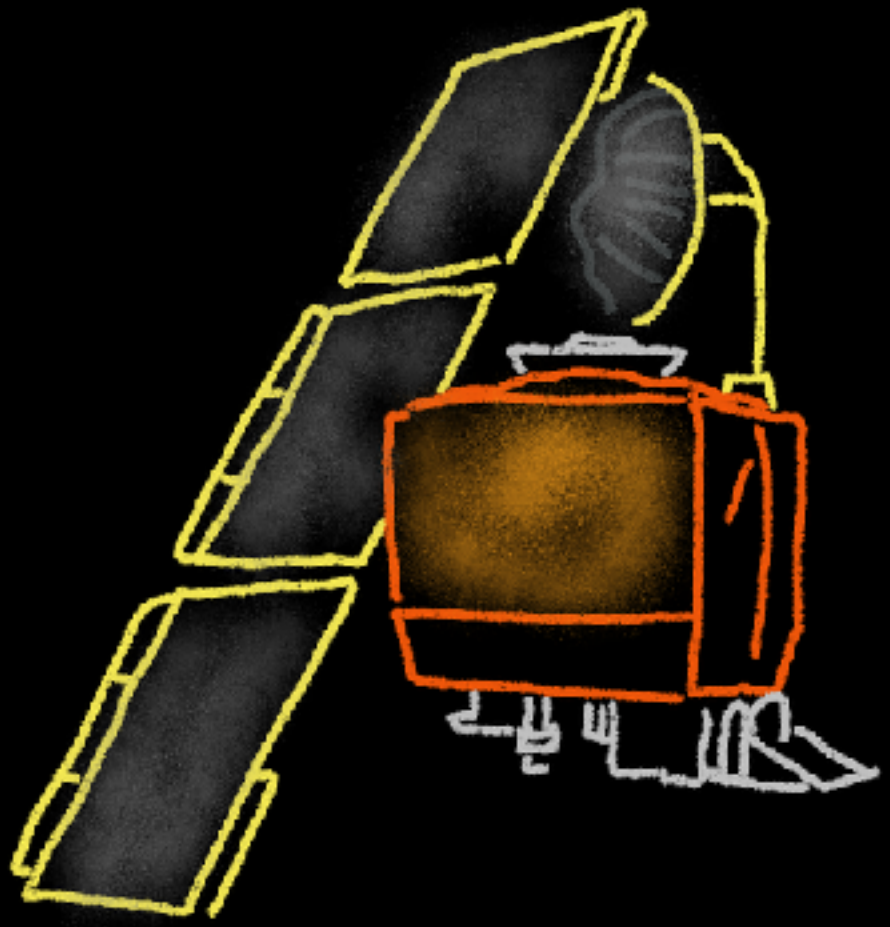
Domain

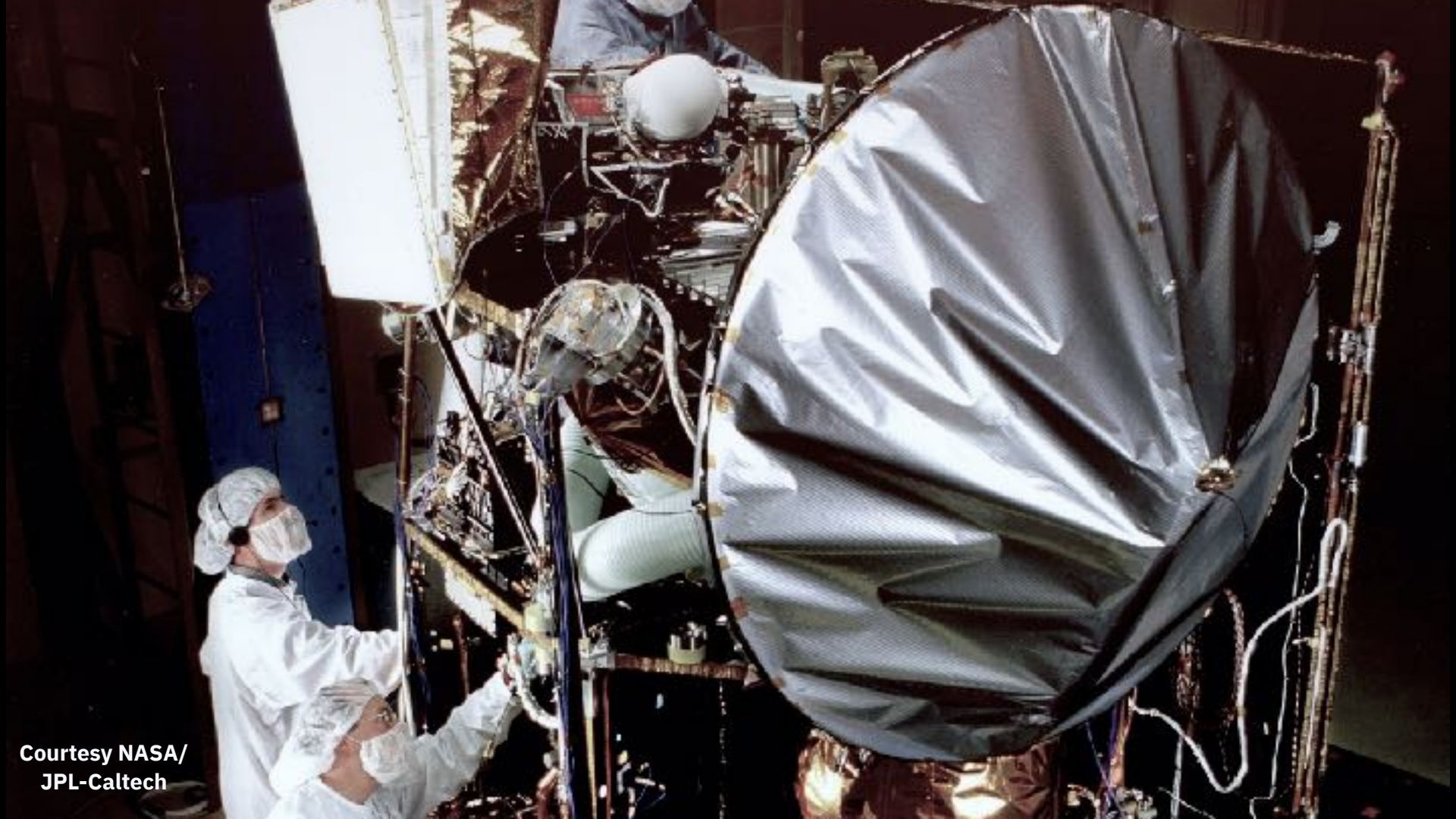


Microservice

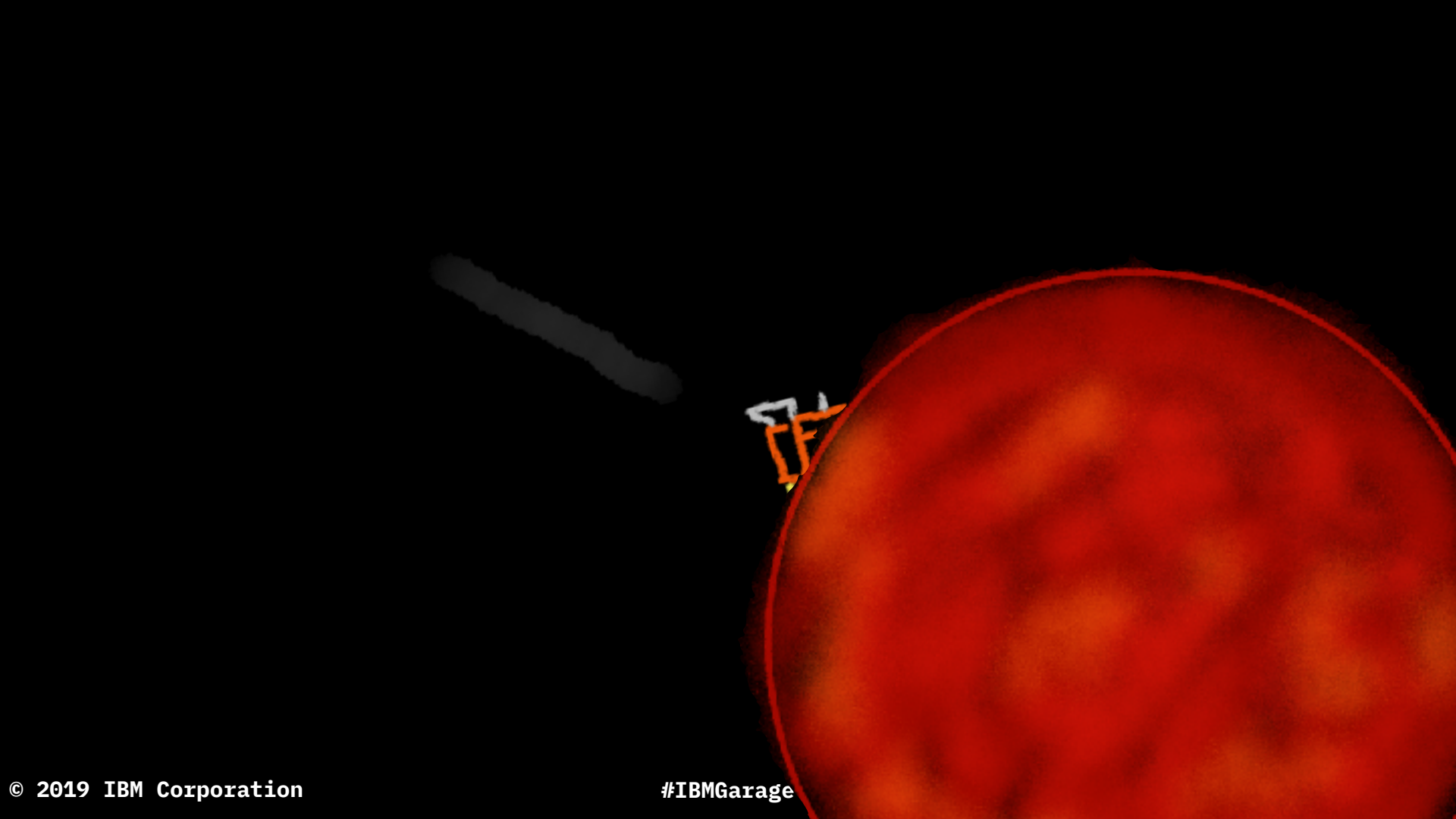
Domain

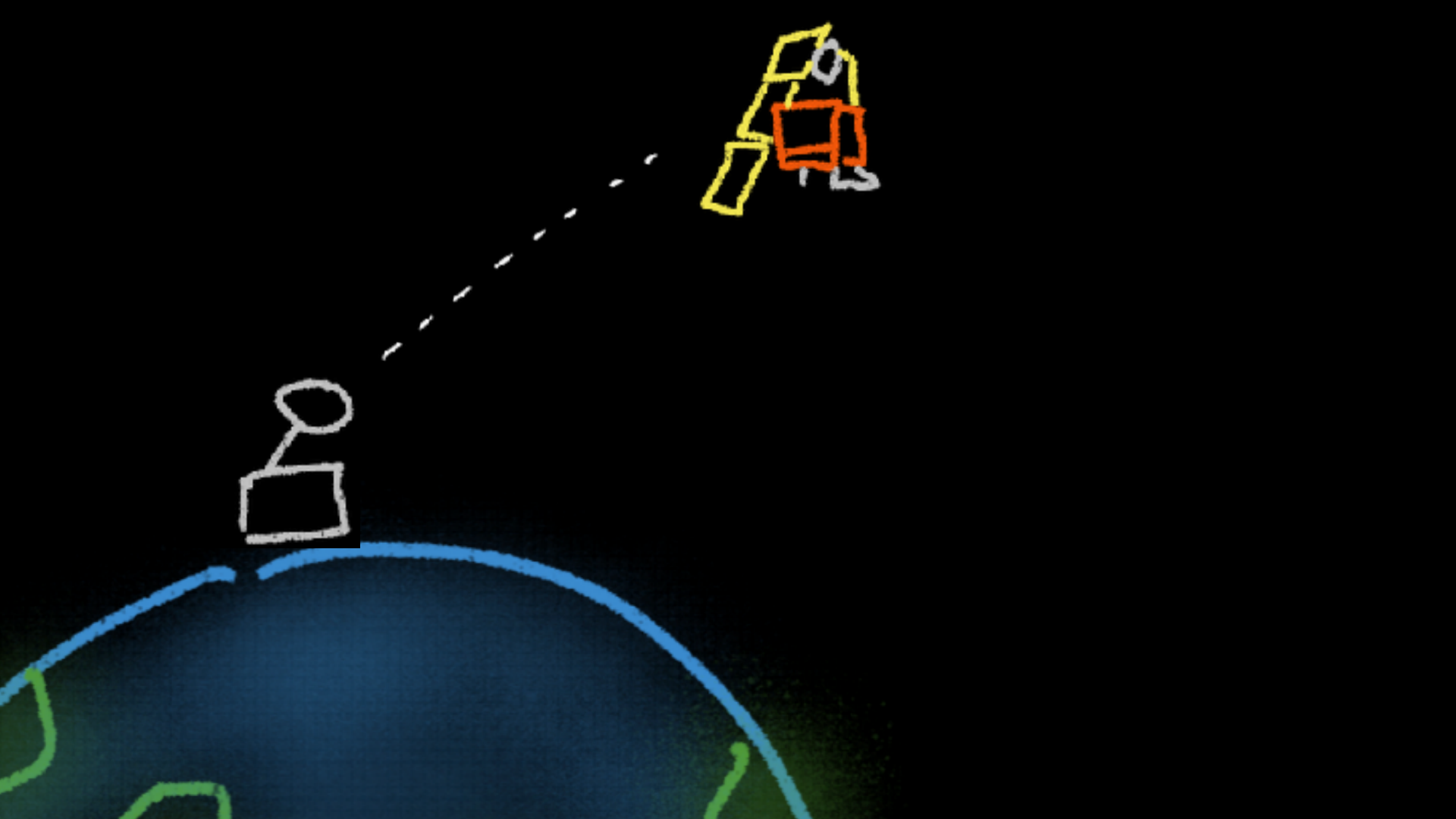


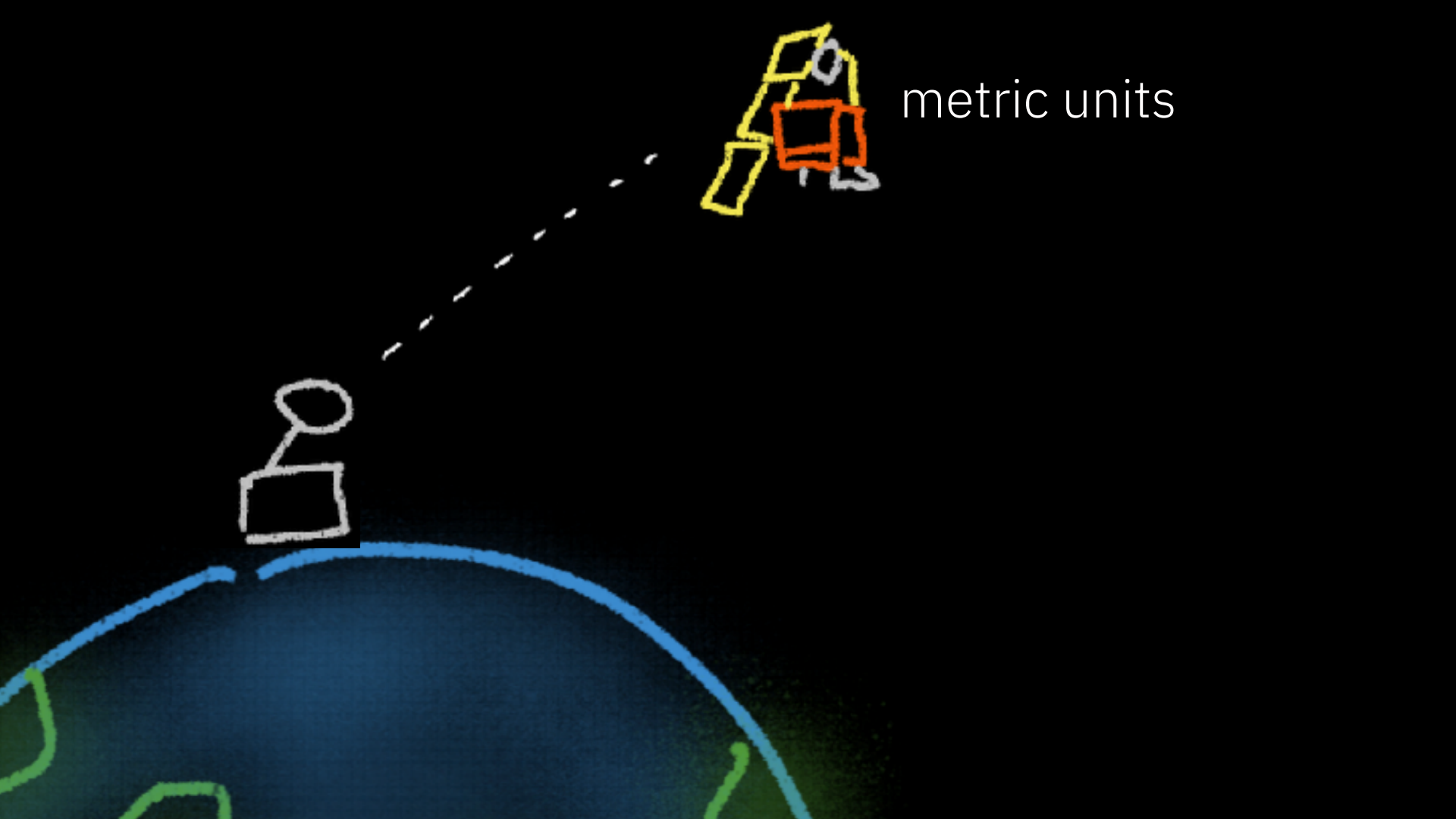


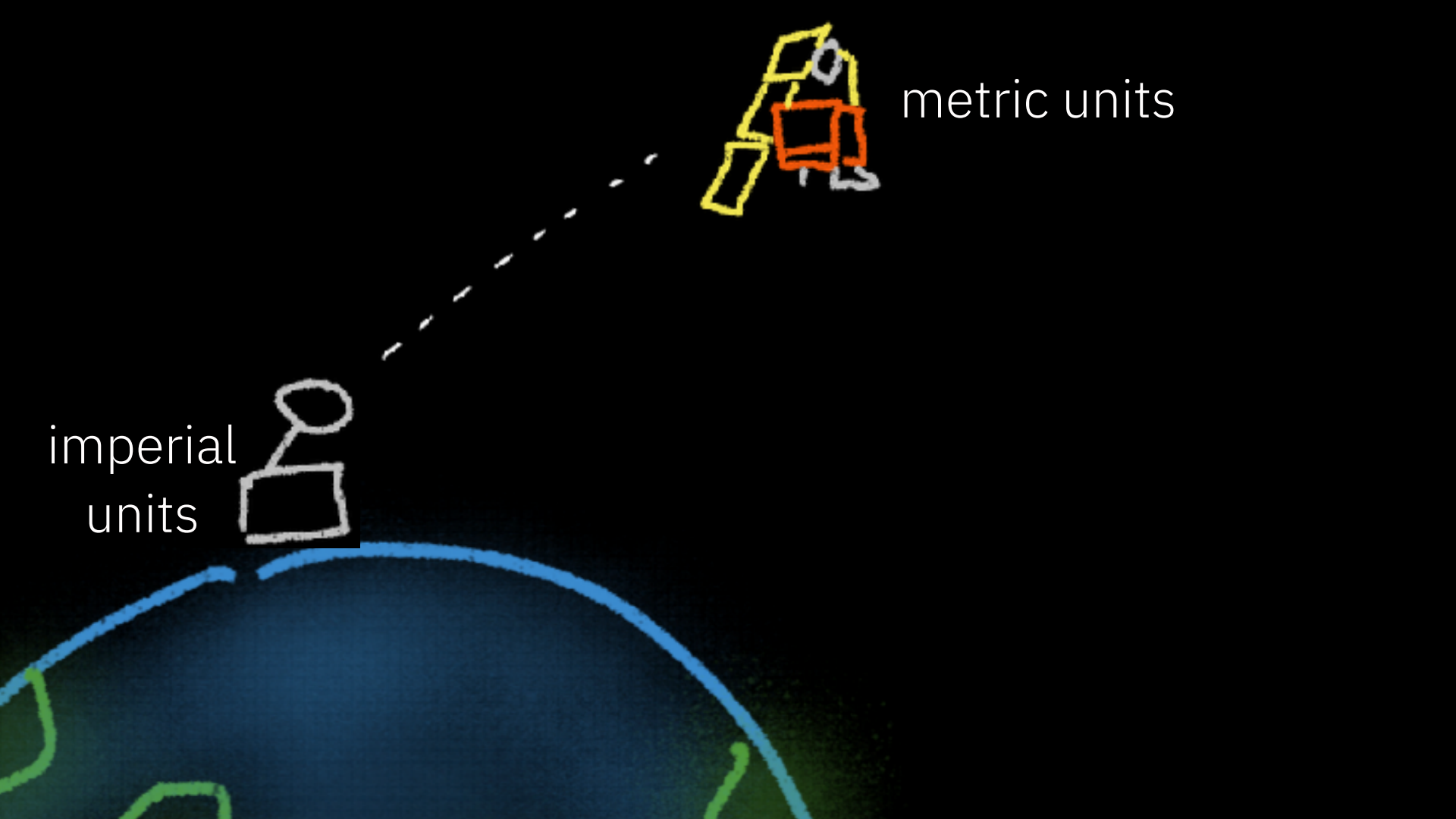


Courtesy NASA/
JPL-Caltech






metric units


metric units
imperial units


metric units
imperial units
distributing did not help

microservices **need**
consumer-driven contract tests

fail

the not-actually-continuous
continuous integration and
continuous deployment

“we have a CI/CD”

CI/CD is something you **do**
not a tool you buy

“i’ll merge my branch into our CI next week”

“CI/CD … CI/CD … CI/CD …

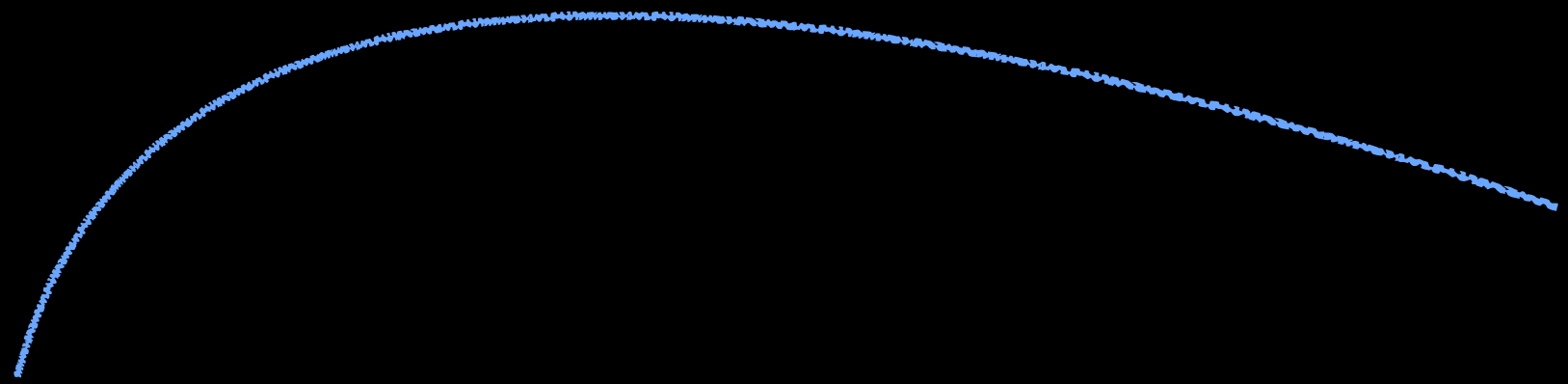
we release every six months …

CI/CD …. ”

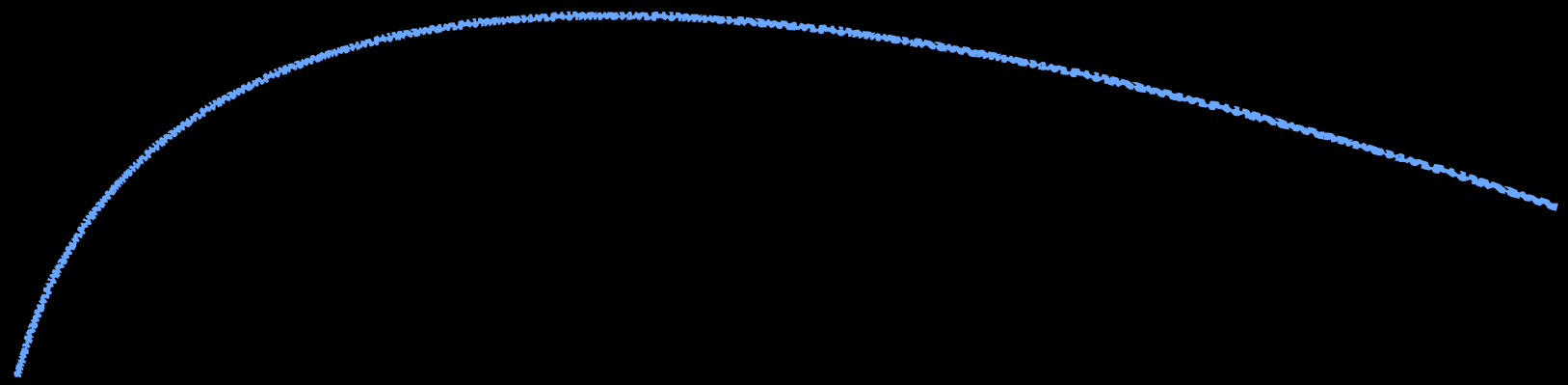
continuous.

I don’t think that word means
what you think it means.

how often should you push to master?

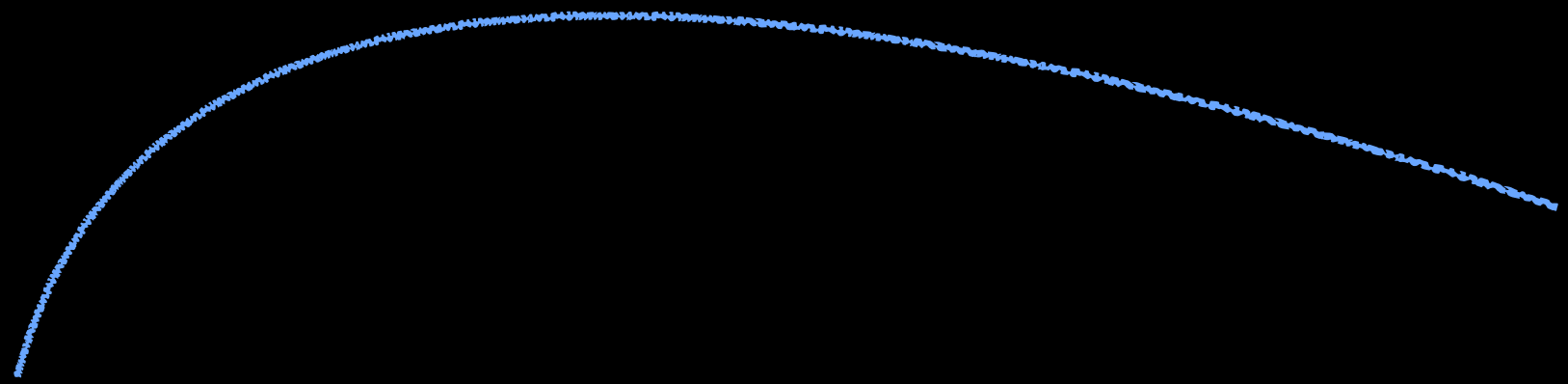


how often should you integrate?



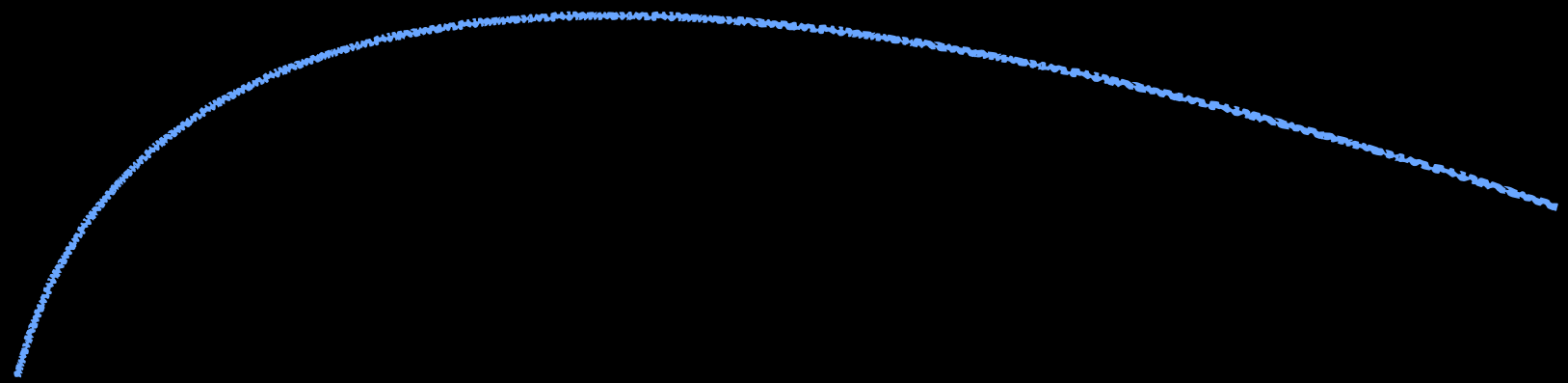
how often should you integrate?

every character



how often should you integrate?

every character

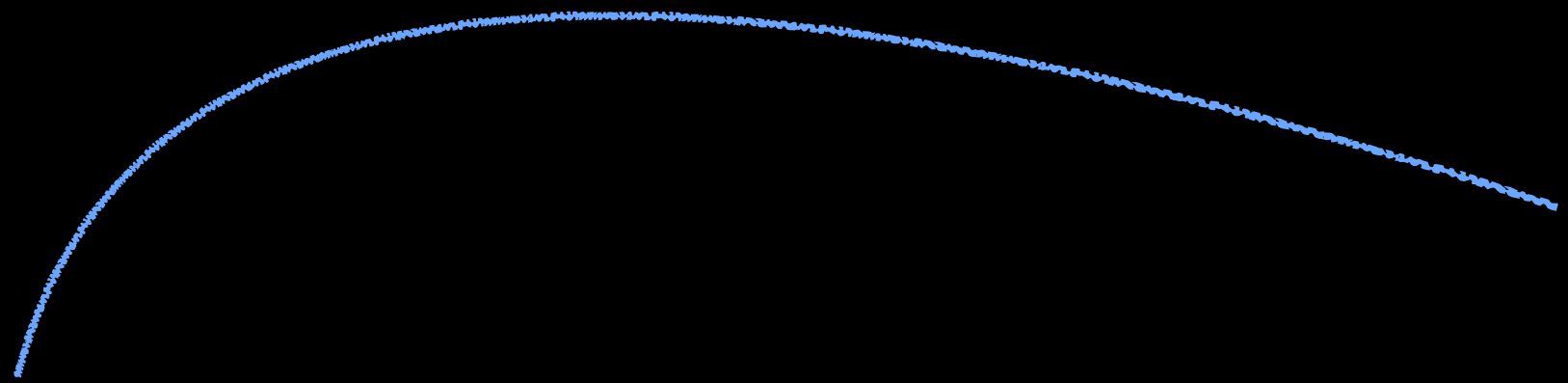


actually continuous
... but stupid

how often should you integrate?

every character

every commit
(several times an hour)



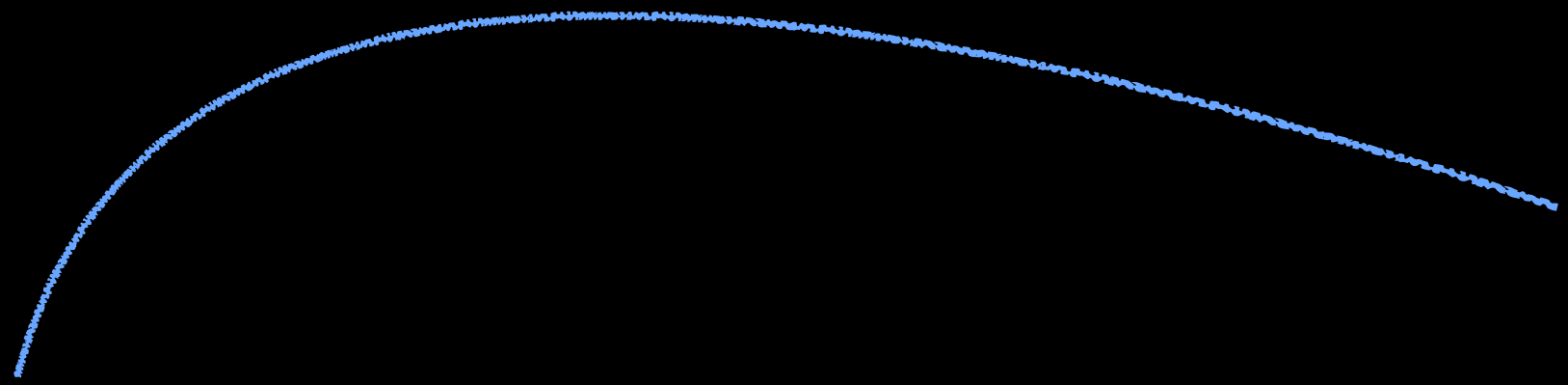
actually continuous
... but stupid

how often should you integrate?

every few commits
(several times a day)

every character

every commit
(several times an hour)



actually continuous
... but stupid

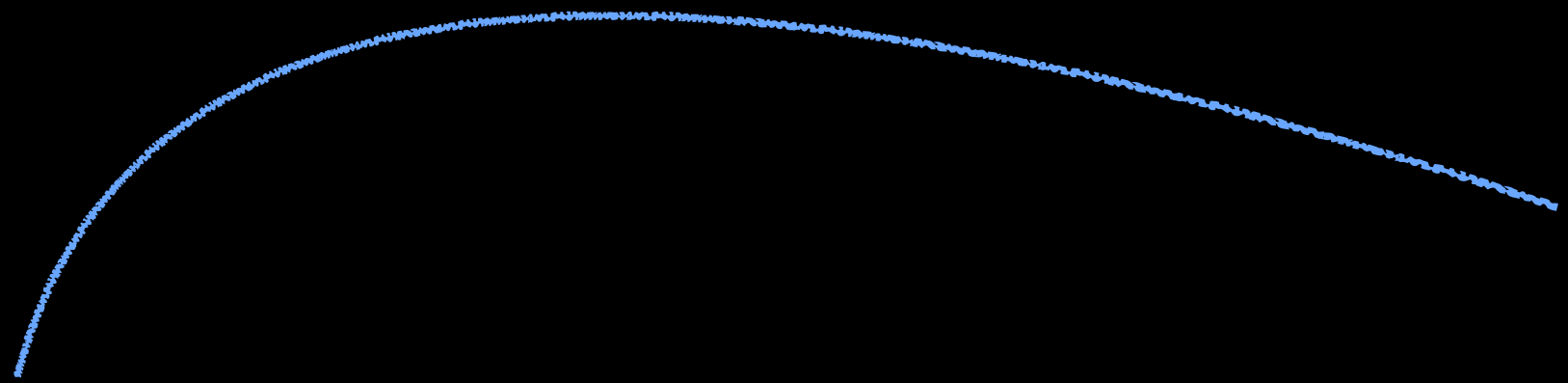
how often should you integrate?

every few commits
(several times a day)

every character

every commit
(several times an hour)

once a day



actually continuous
... but stupid

how often should you integrate?

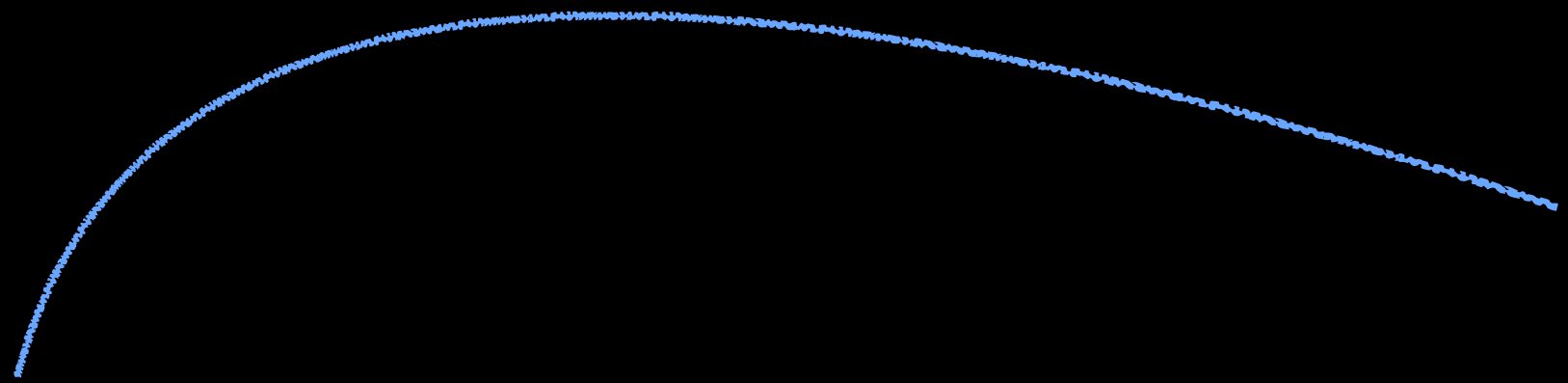
every few commits
(several times a day)

every character

every commit
(several times an hour)

once a day

once a
week



actually continuous
... but stupid

how often should you integrate?

every few commits
(several times a day)

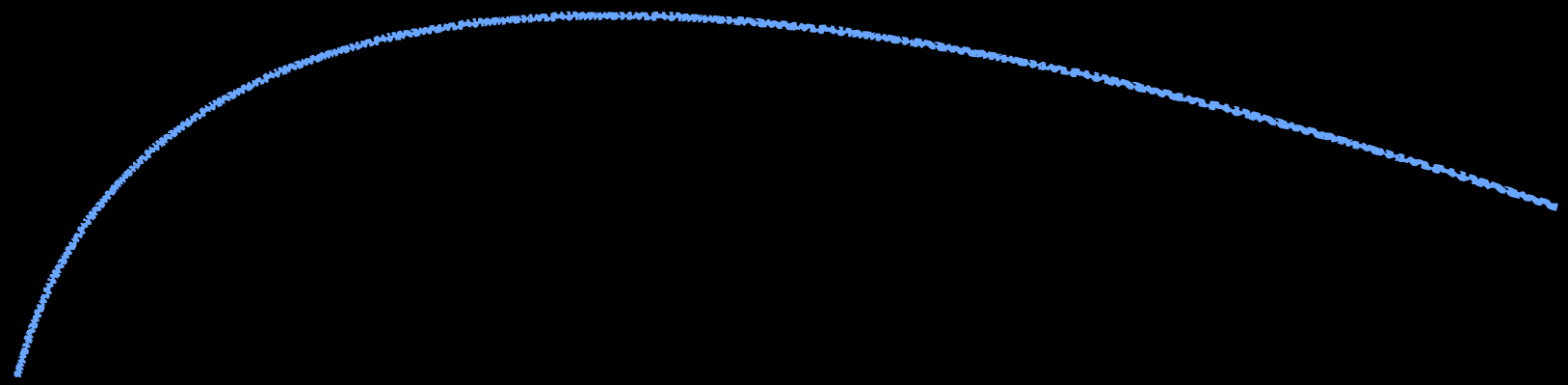
every character

every commit
(several times an hour)

once a day

once a
week

once a
month



actually continuous
... but stupid

how often should you integrate?

every few commits
(several times a day)

every character

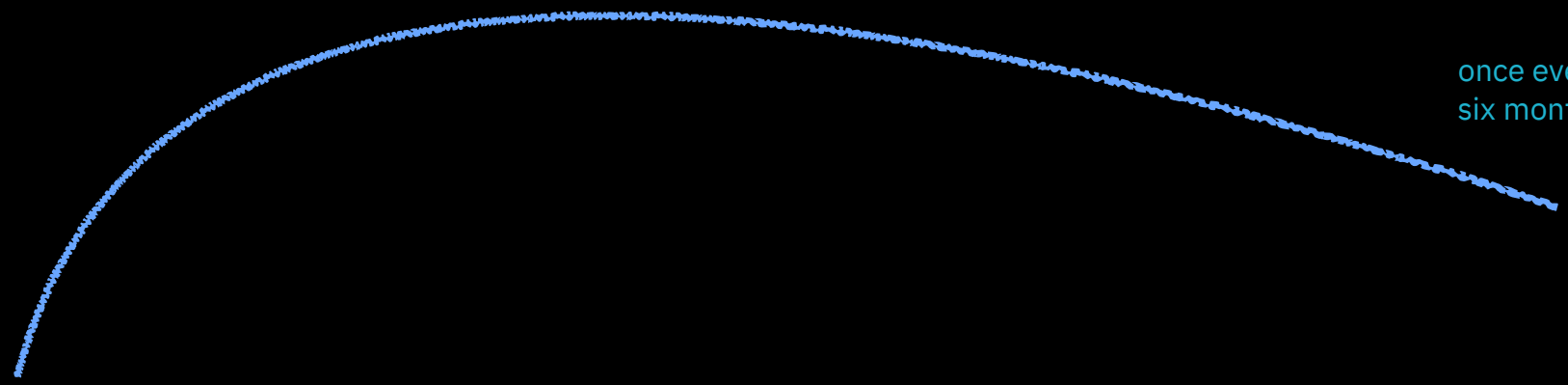
every commit
(several times an hour)

once a day

once a
week

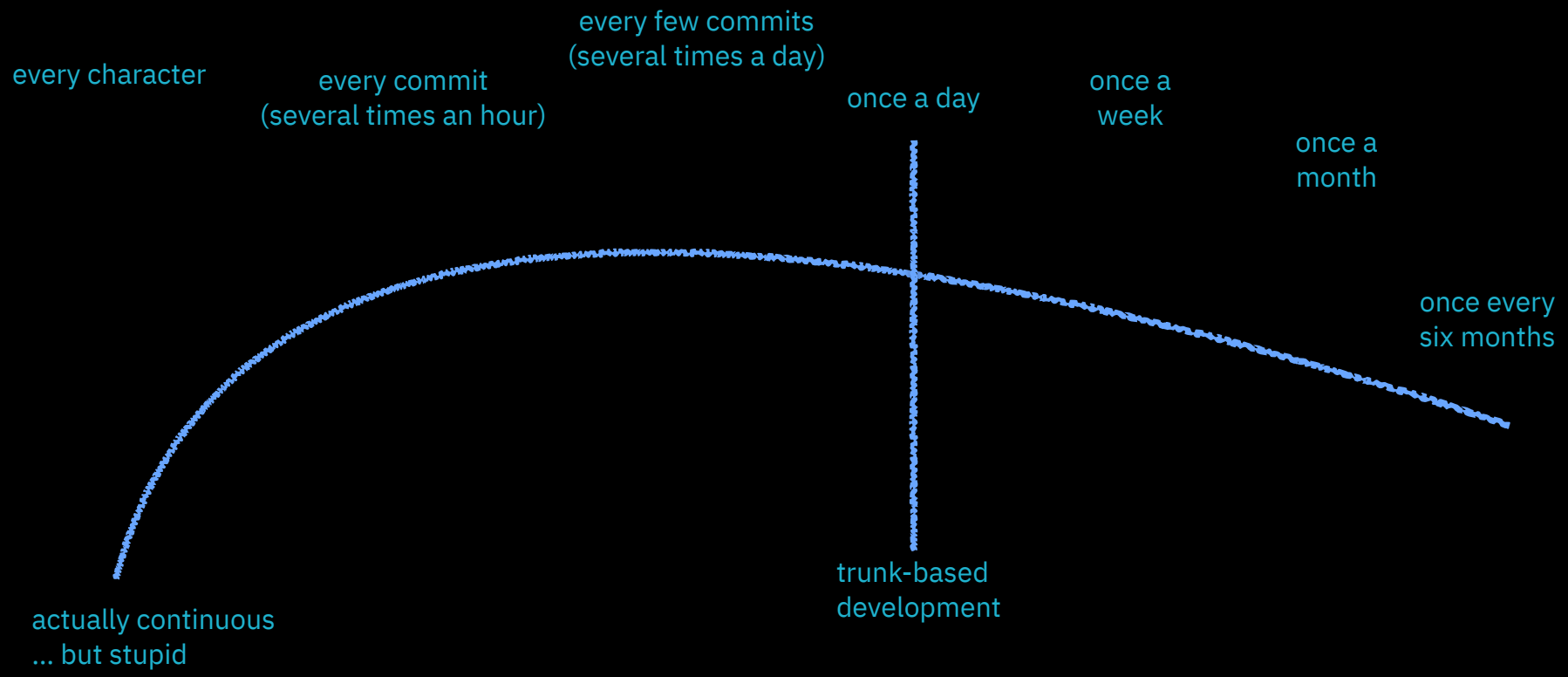
once a
month

once every
six months

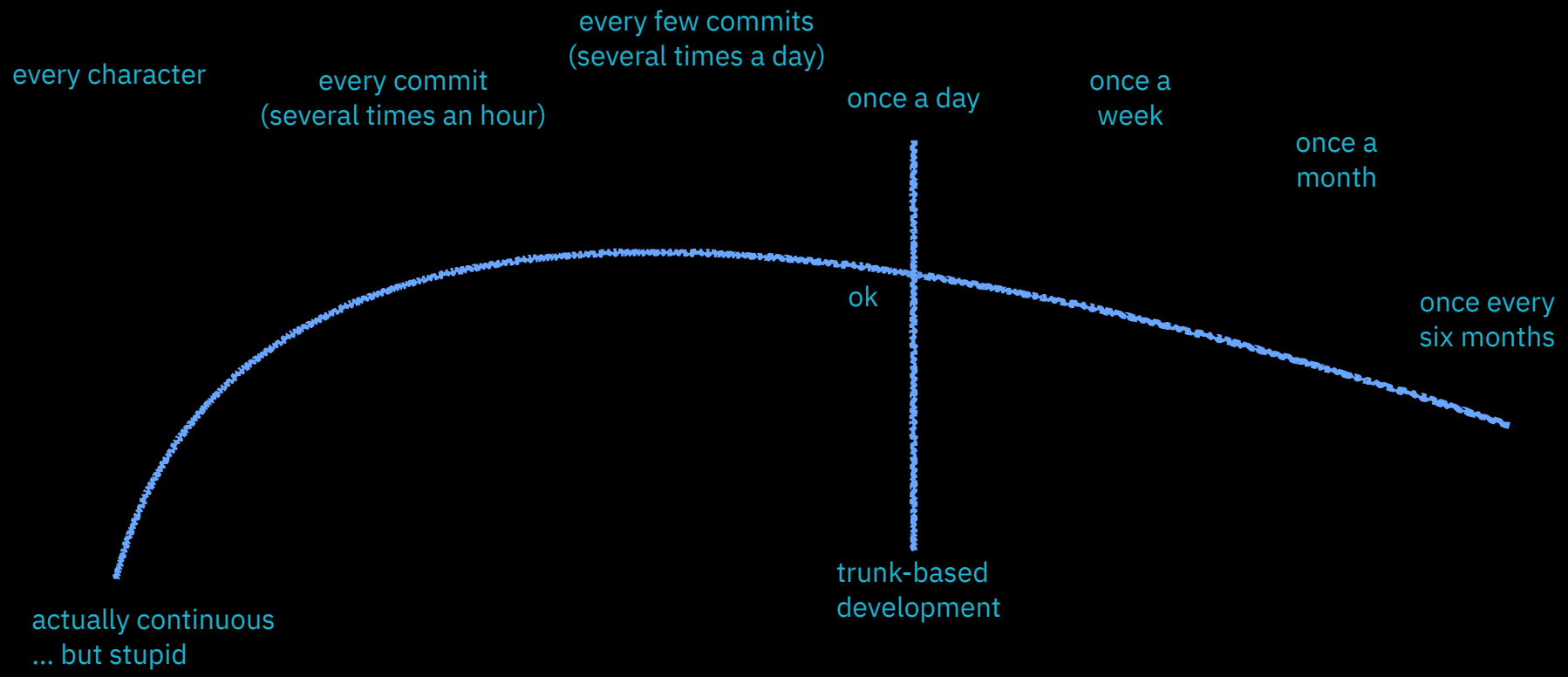


actually continuous
... but stupid

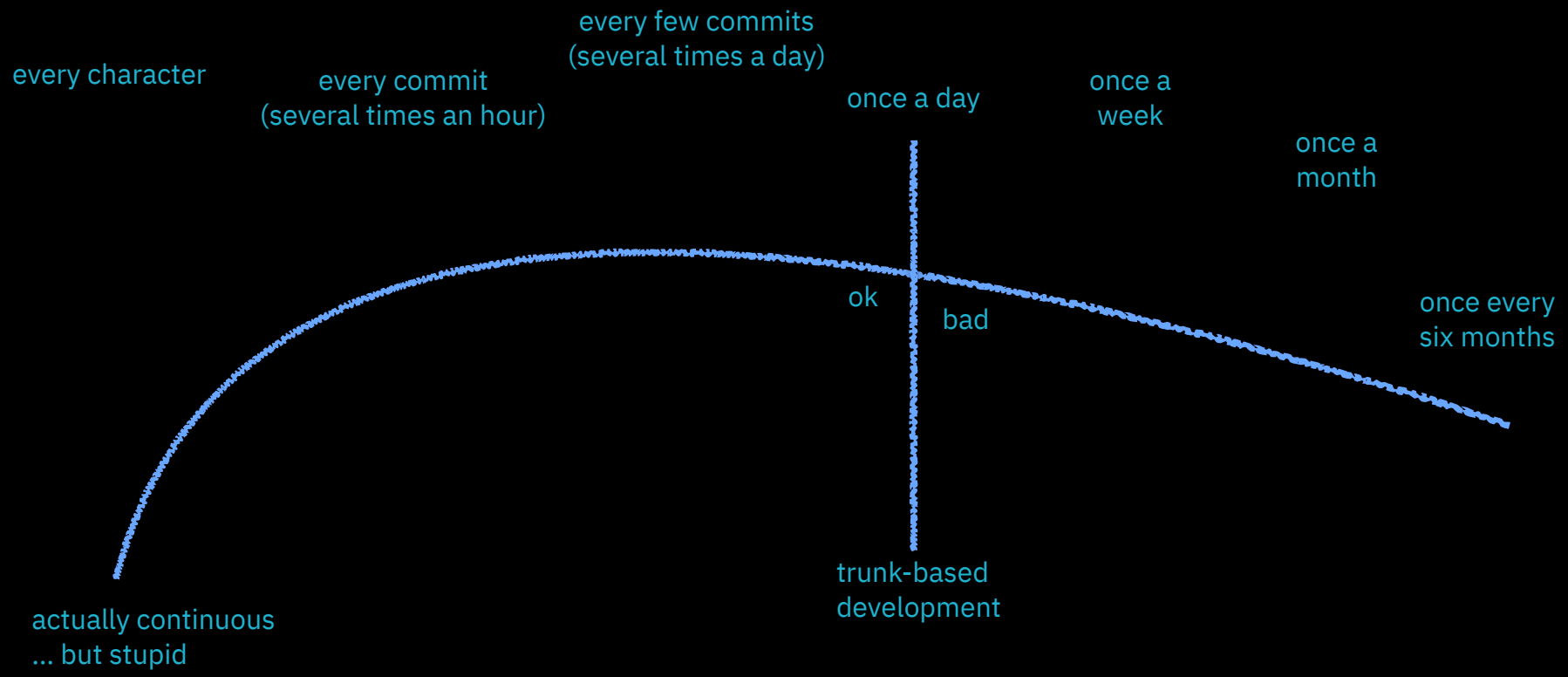
how often should you integrate?



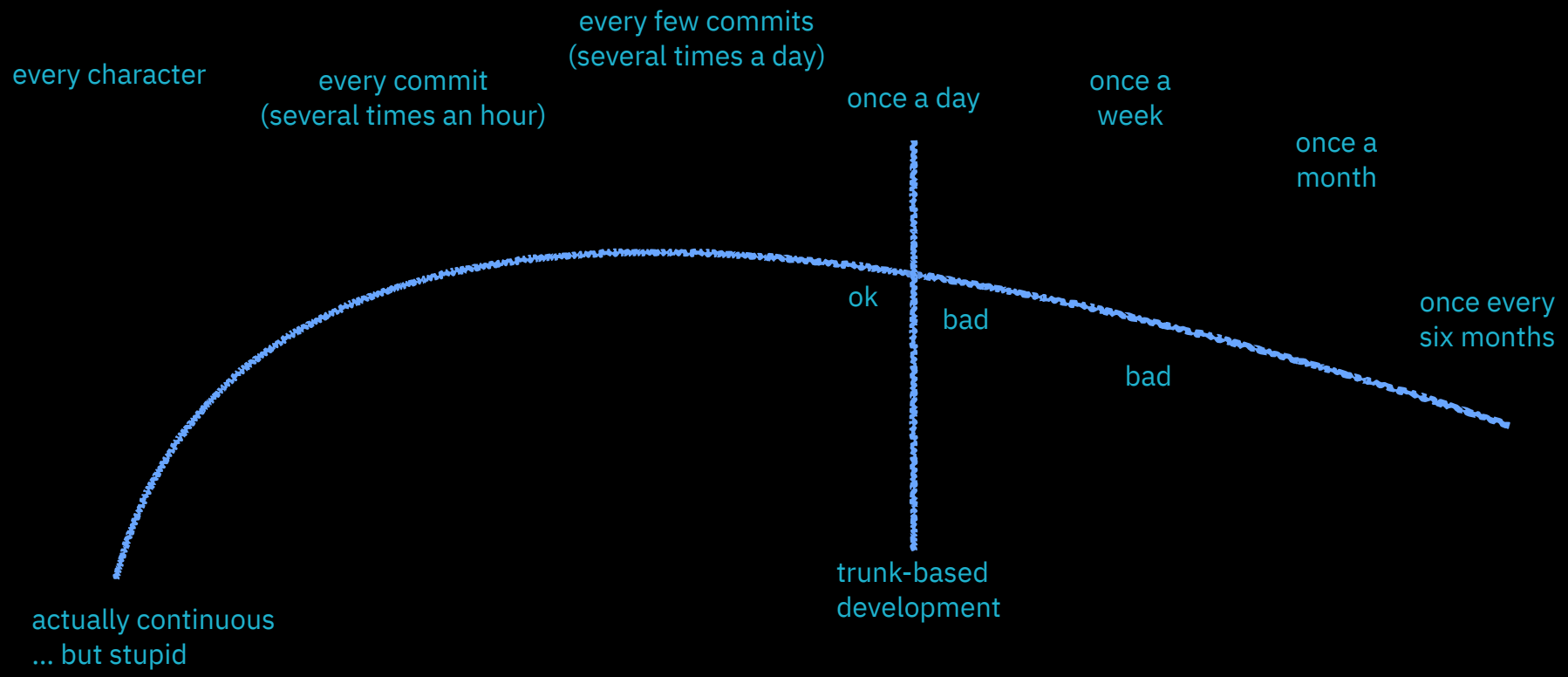
how often should you integrate?



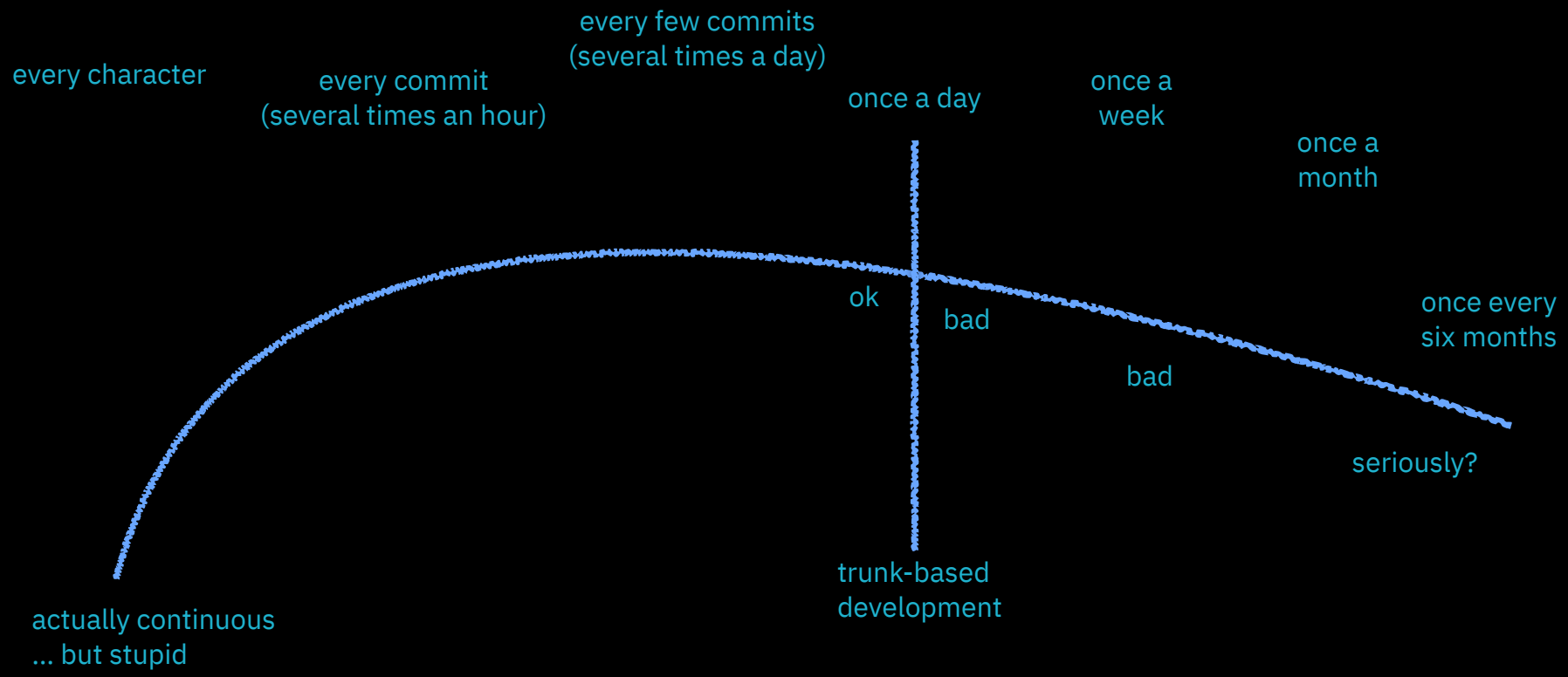
how often should you integrate?



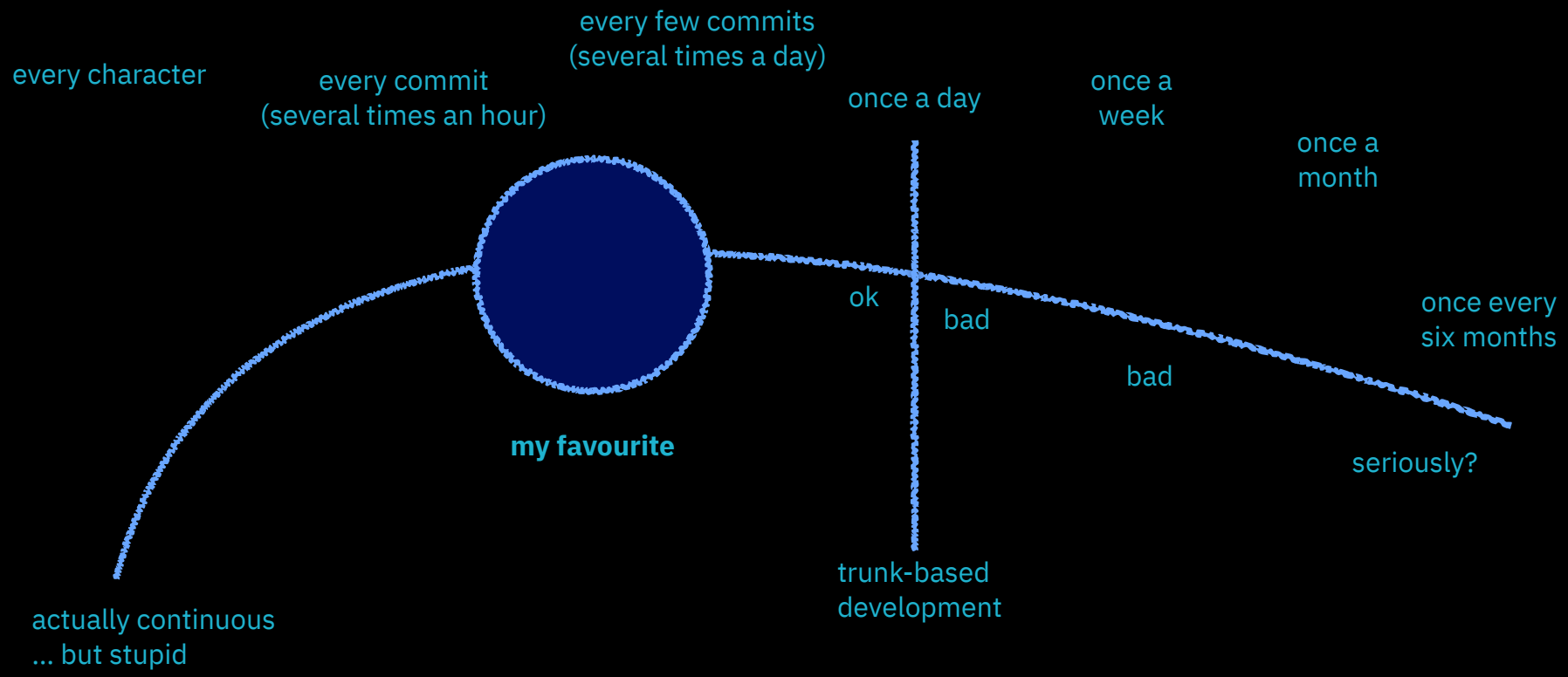
how often should you integrate?



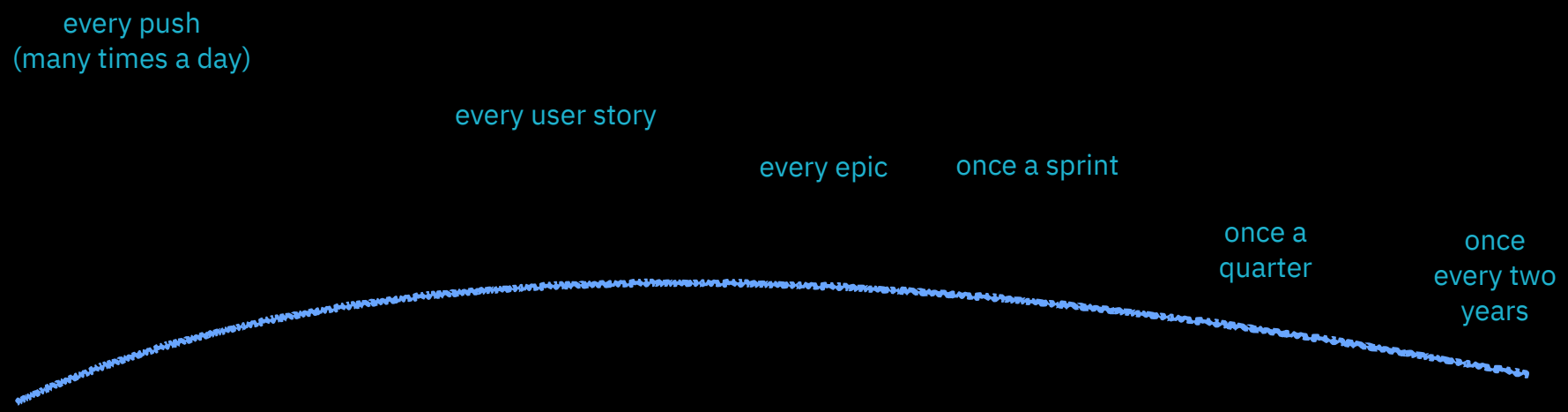
how often should you integrate?



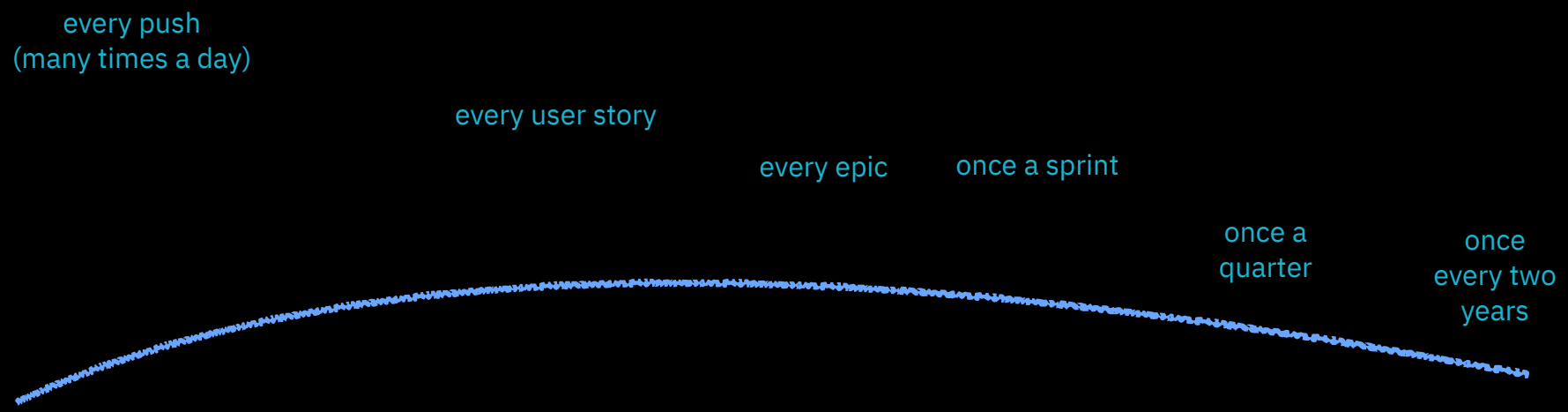
how often should you integrate?



how often should you release?



how often should you deploy?



how often should you deploy?

every push
(many times a day)

every user story

every epic

once a sprint

once a
quarter

once
every two
years

(need a good handle on
feature flags)

how often should you deploy?

every push
(many times a day)

every user story

every epic

once a sprint

ok

once a
quarter

once
every two
years

(need a good handle on
feature flags)

how often should you deploy?

every push
(many times a day)

every user story

every epic

once a sprint

once a
quarter

once
every two
years

ok

old-
school

(need a good handle on
feature flags)

how often should you deploy?

every push
(many times a day)

every user story

every epic

once a sprint

once a
quarter

once
every two
years

ok

sigh

old-
school

(need a good handle on
feature flags)

how often should you deploy?

every push
(many times a day)

every user story

every epic

once a sprint

once a
quarter

once
every two
years

ok

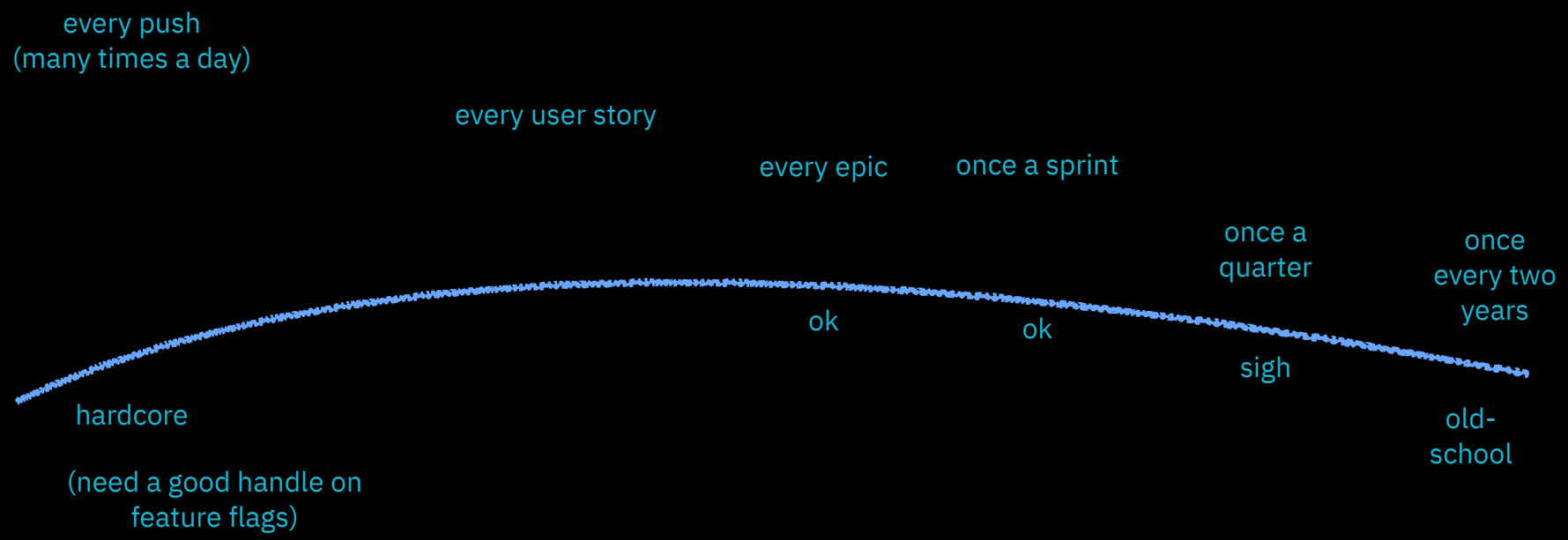
ok

sigh

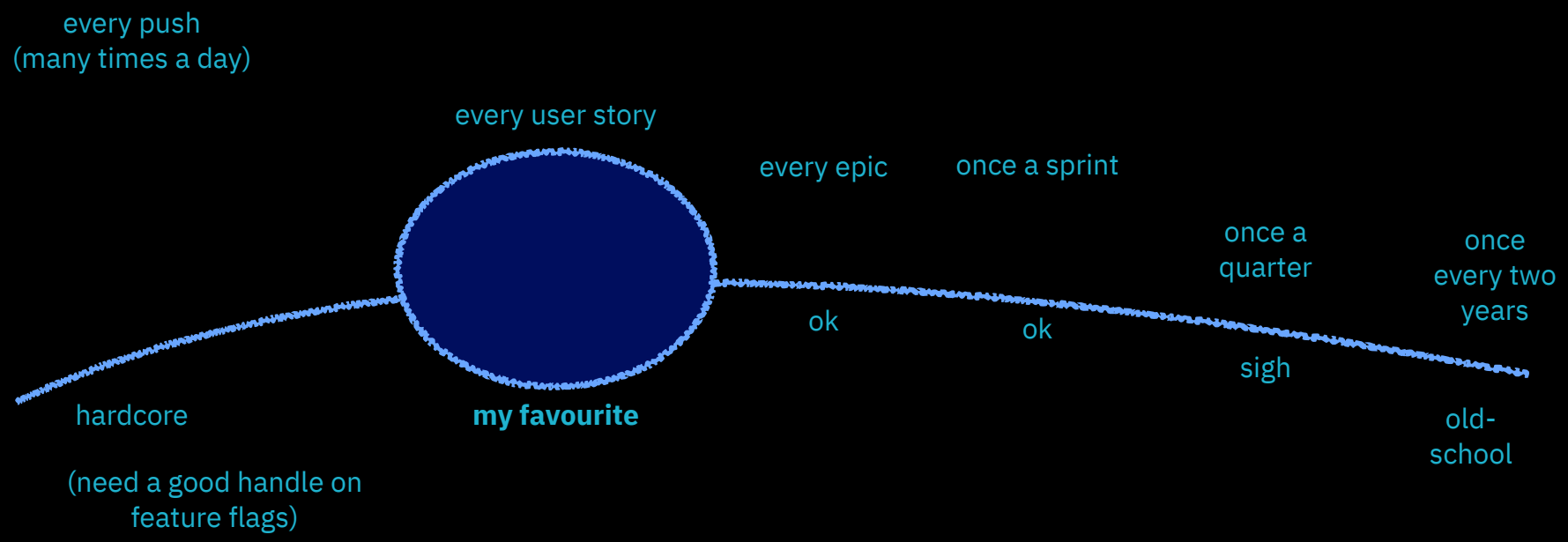
old-
school

(need a good handle on
feature flags)

how often should you deploy?



how often should you deploy?

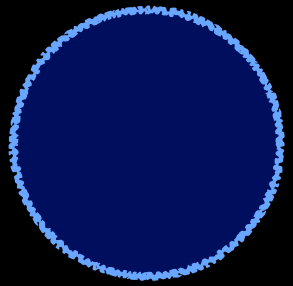


how often should you test in staging?

how often should you deliver?

how often should you deliver?

every push



my favourite

“we can’t actually **release** this.”


why?

what’s stopping more frequent deploys?

“we can’t release this microservice…

we deploy all our microservices at the same time.”

“we can’t ship until every feature is complete”

if you're not embarrassed by your first release it was too late

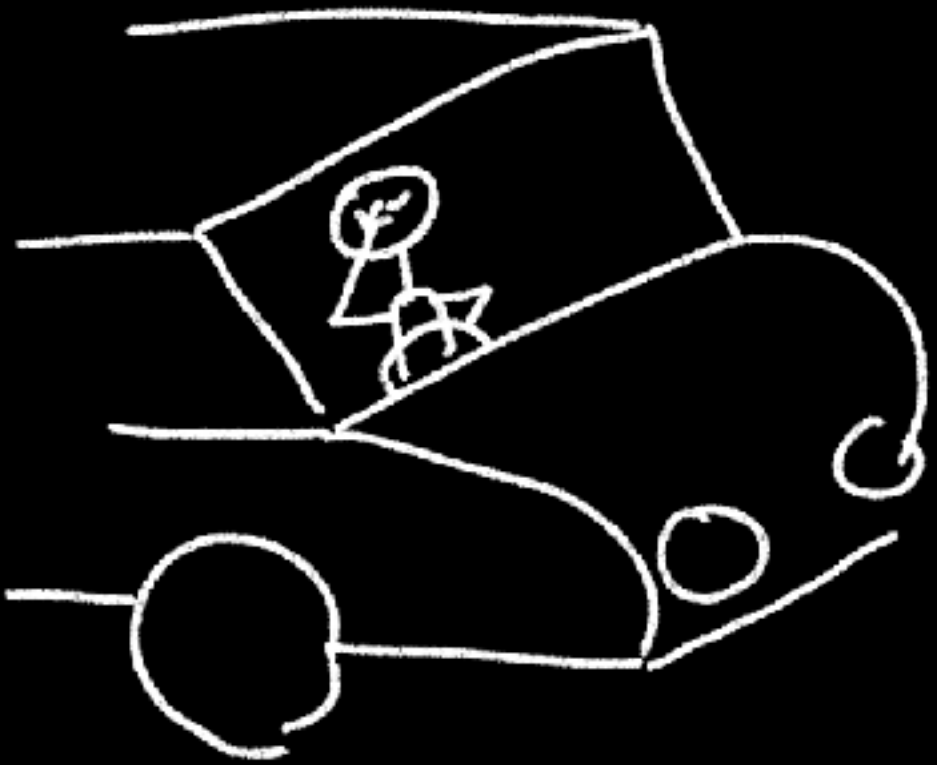
- Reid Hoffman

speed

speed

what’s the point of architecture that can go faster, if you don’t go faster?

what’s the point of architecture that can go faster, if you don’t go faster?



drive a car

feedback is good engineering

feedback is good business

deferred wiring

feature flags

A/B testing

canary deploys

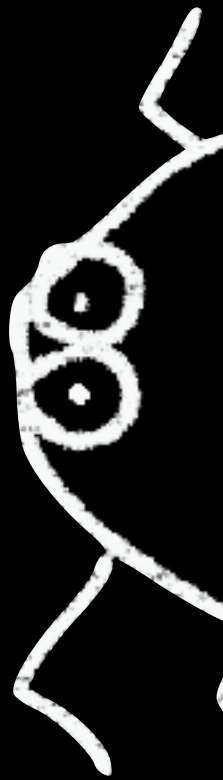
fail

the
‘someday’
automation

“our tests aren’t automated”

“we don’t know if our code works”

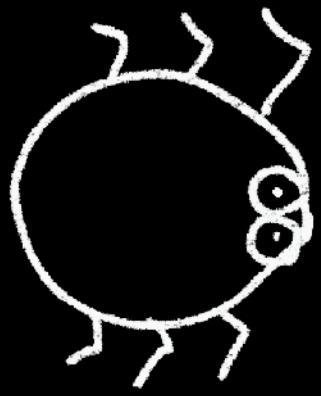
“we don’t know if our code works”



systems **will** behave in unexpected ways

dependency updates can change behaviour

“we can’t ship until we have more confidence in the quality”

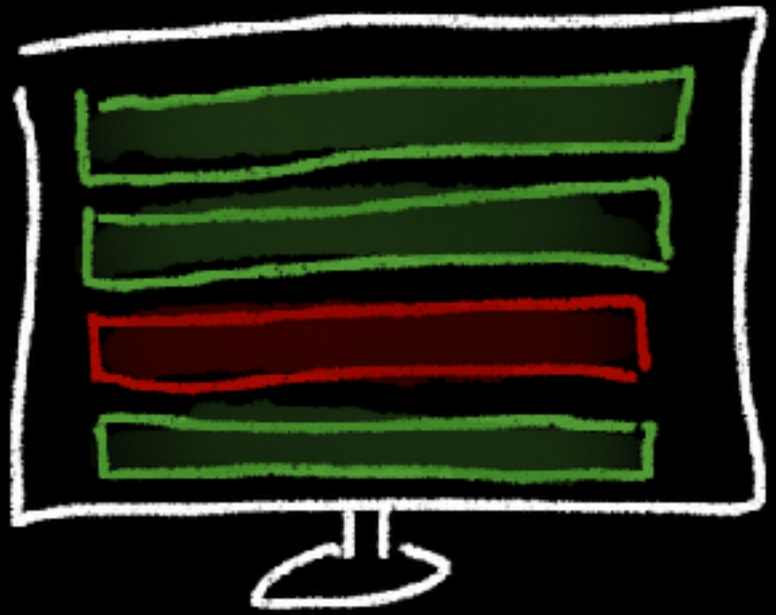


microservices **need**
automated integration tests

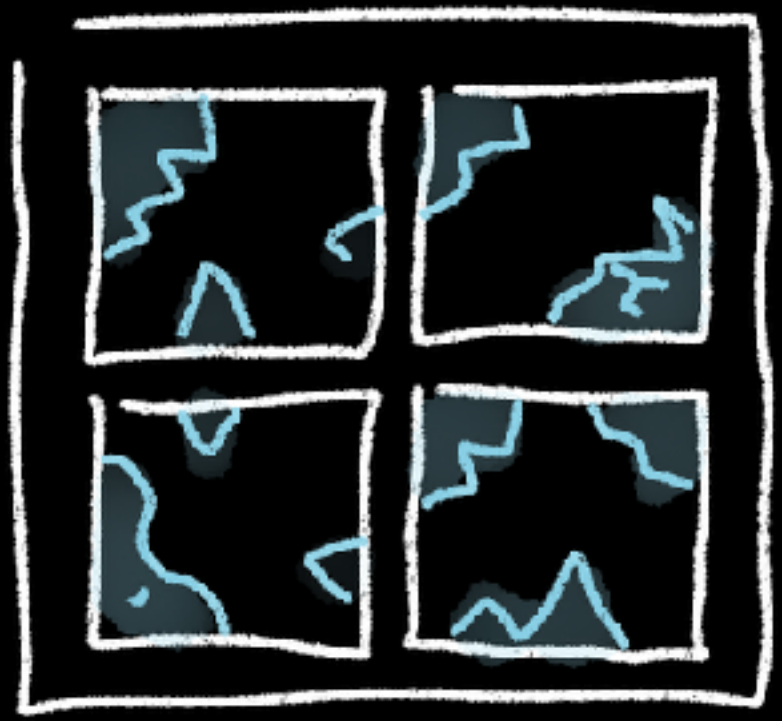
not a good CI/CD indicator

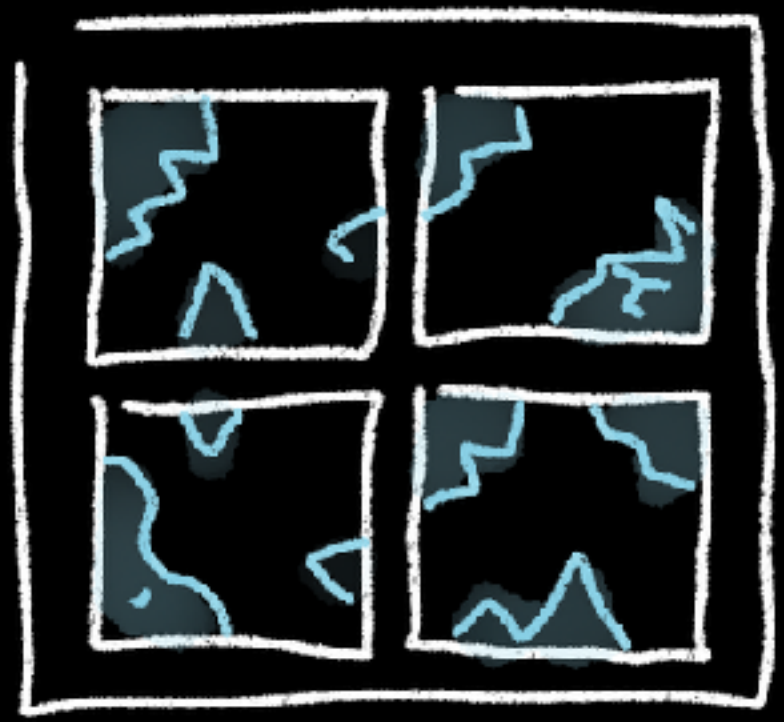
a good CI/CD indicator

“we don’t know when the build is broken”



a good build radiator

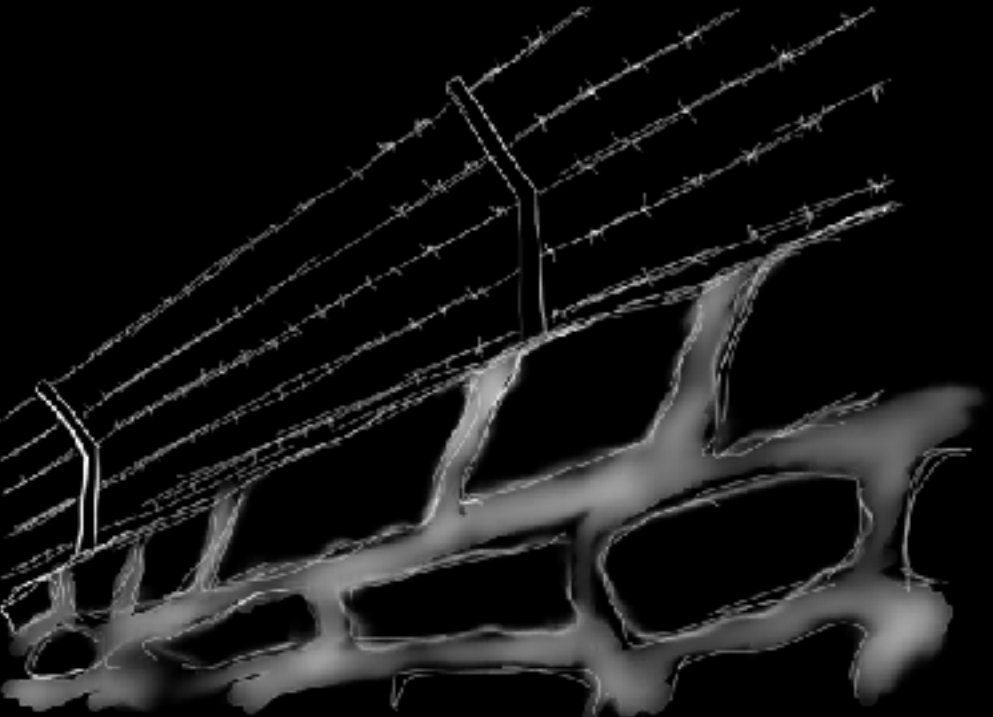




“oh yes, that build has been broken for a few weeks...”

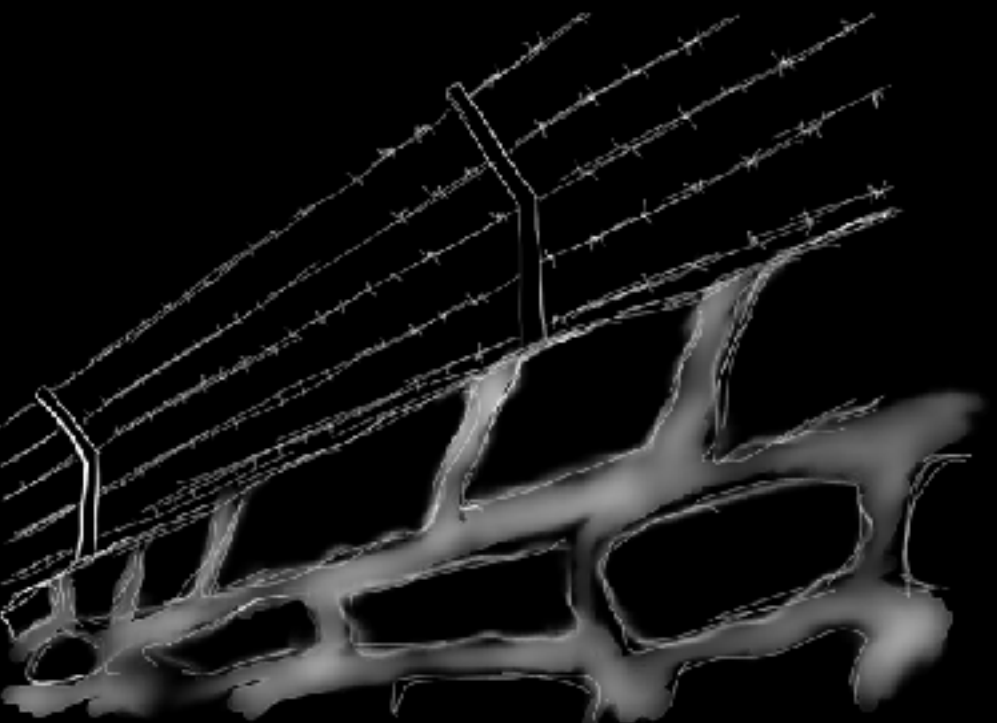
fail

the locked-
down totally
rigid inflexible
un-cloudy cloud

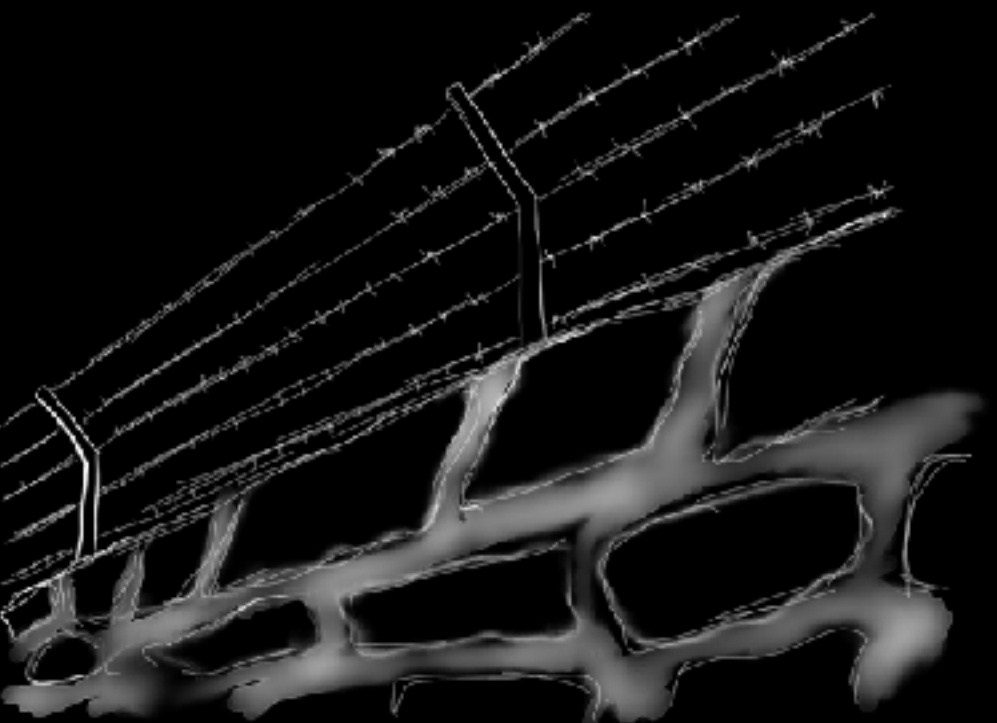


@holly_cummins

“we’ve configured our network!



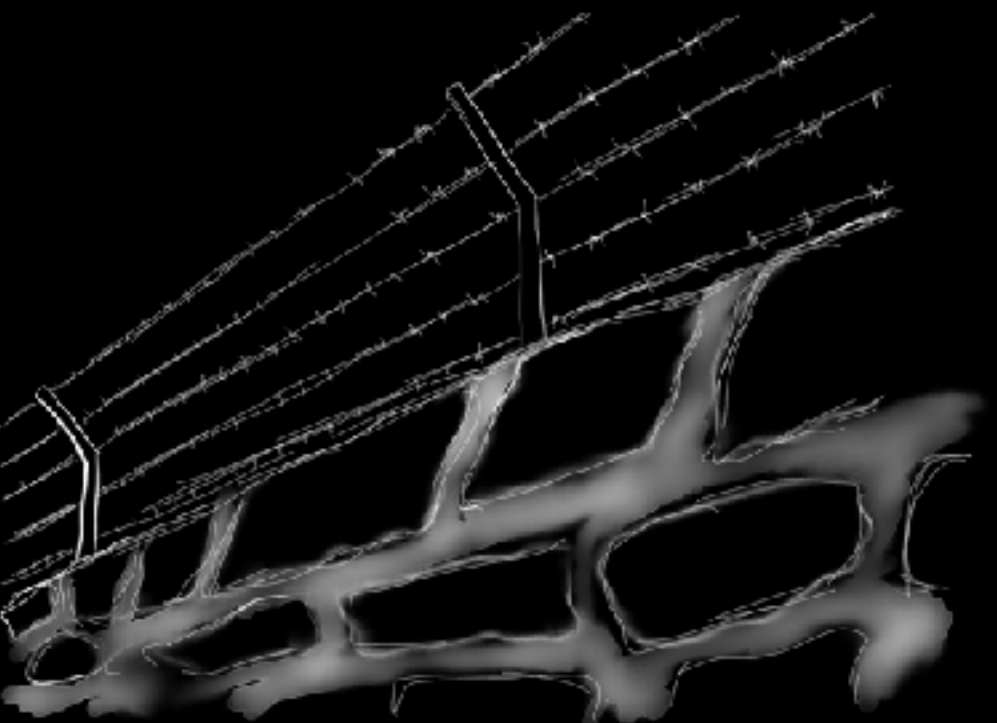
@holly_cummins



“we’ve configured our network!

you can either access the cloud servers ... or access jira.

@holly_cummins



“we’ve configured our network!

you can either access the cloud servers ... or access jira.

to access both you’d need two machines.”

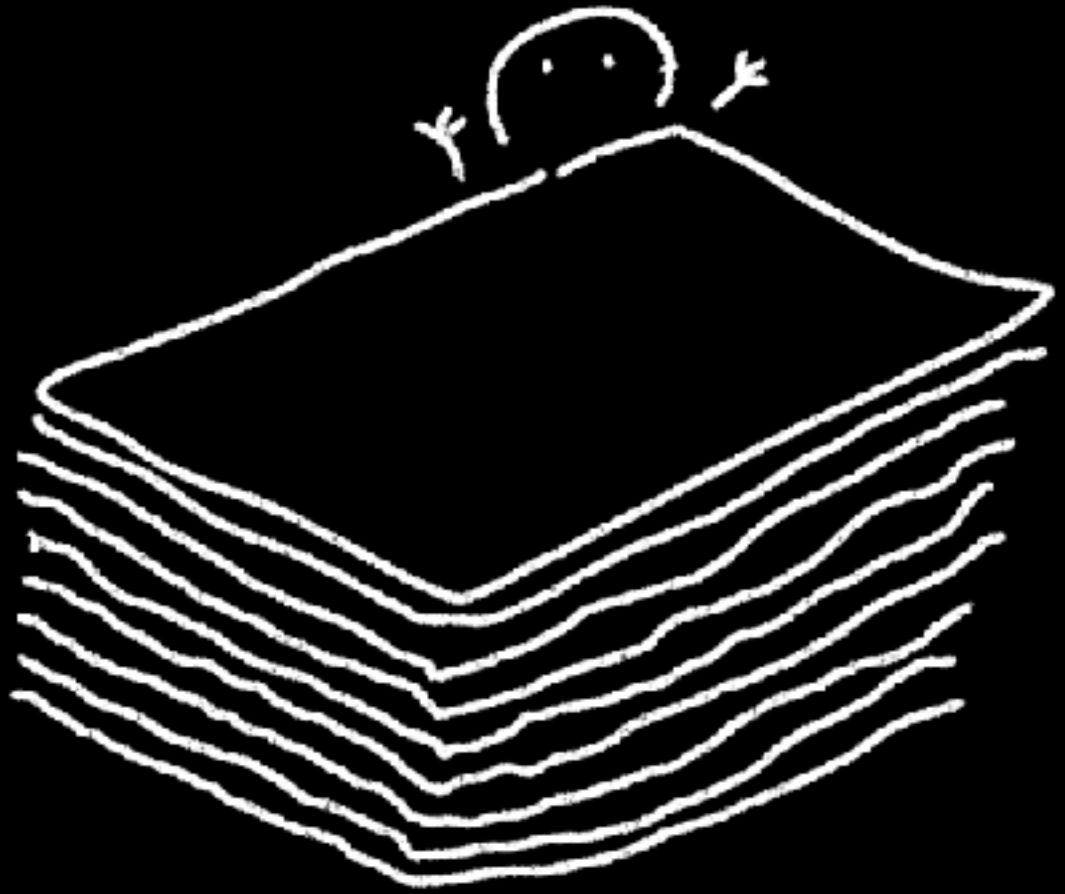
@holly_cummins

“it takes us a week to start coding.”

“it takes us a week to start coding.”

“two days to get a repo ...

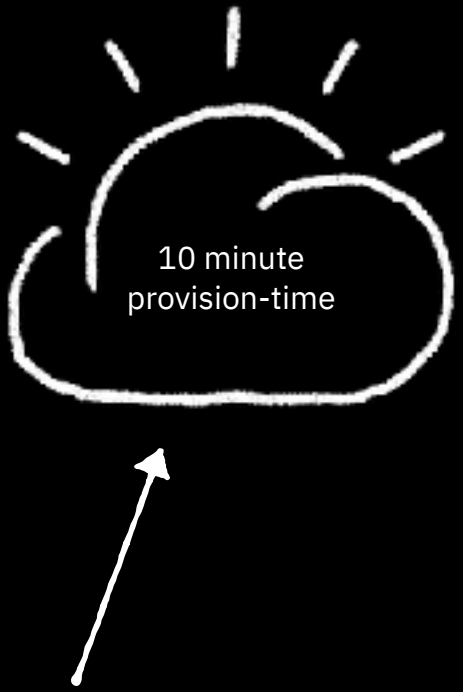
two days to get a pipeline ...”



“we’ve scheduled the architecture board review for a month after the project is ready to ship”

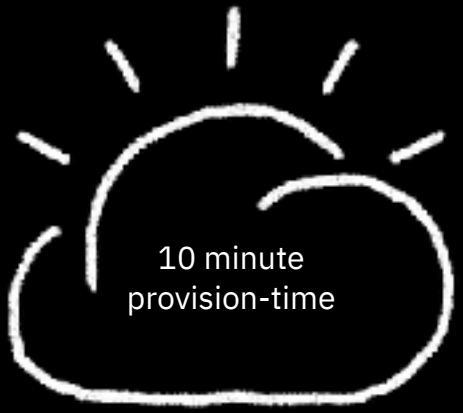
@holly_cummins

“this provisioning software is broken”

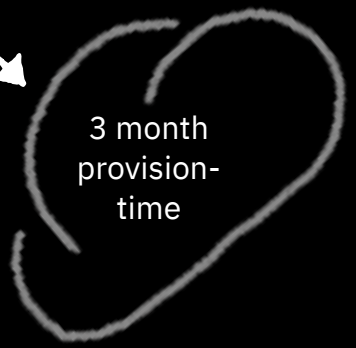


what we sold

“this provisioning software is broken”

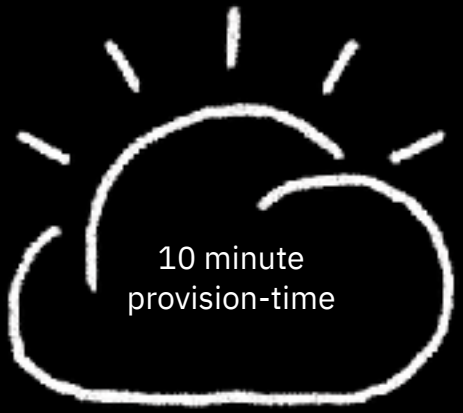


what the client thought they'd got



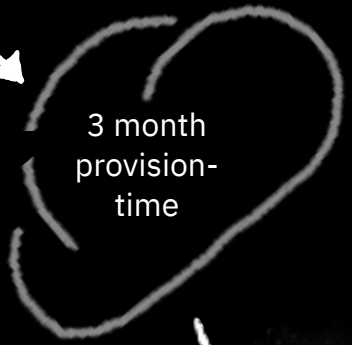
what we sold

"this provisioning software is broken"



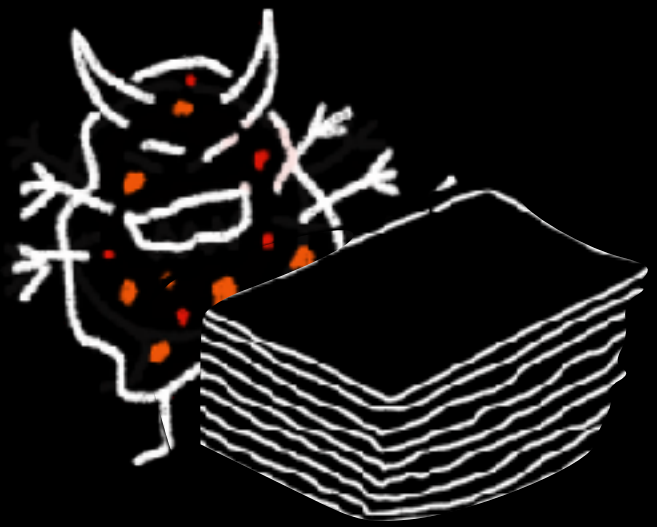
what we sold

what the client thought they'd got

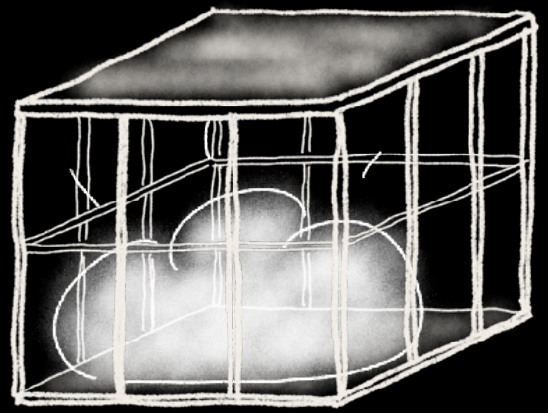


the reason

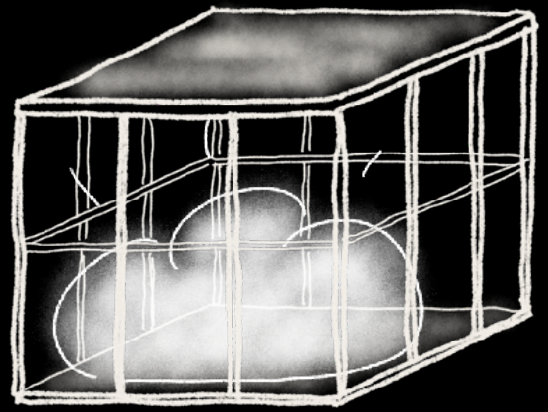
"this provisioning software is broken"



old-style governance isn’t going to work



Provider A



Provider A

Provider B

“we’re going to change cloud provider to fix our procurement process!”

Provider A

Provider B

“we’re going to change cloud provider to fix our procurement process!”

if the developers are the only ones changing, cloud native is not going to work

there is a **cost**:

developers leave

fail

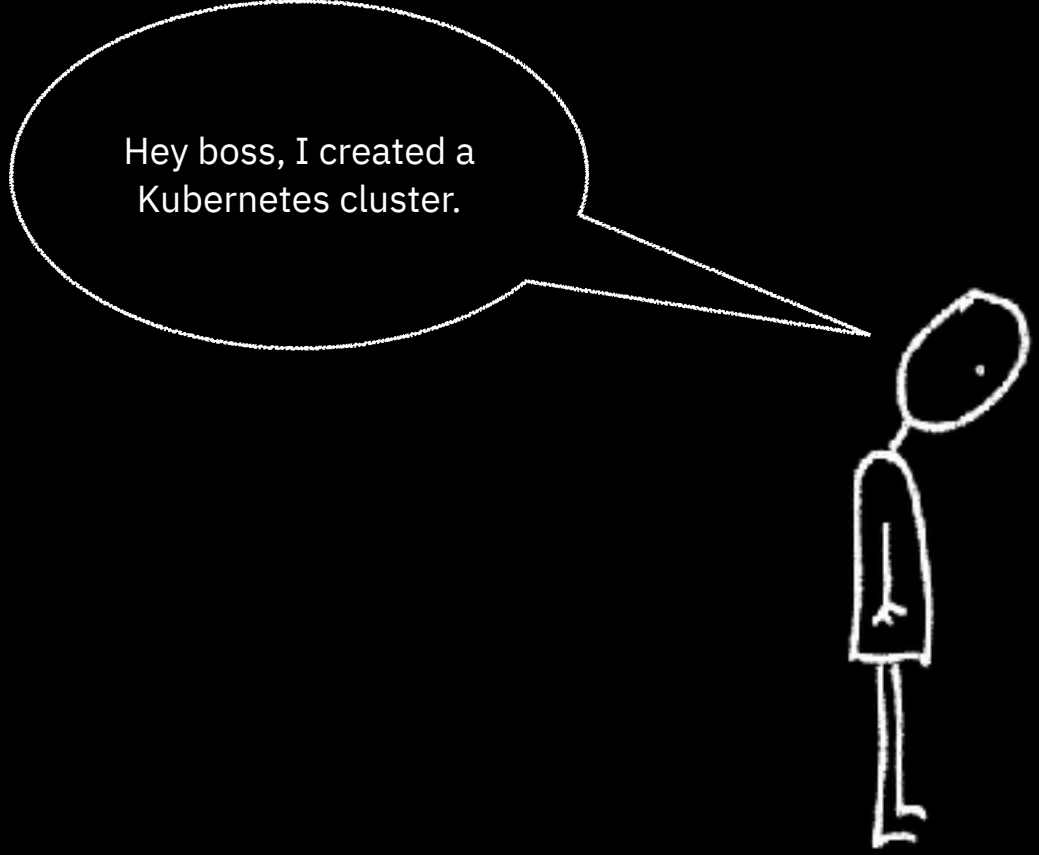
the mystery money pit

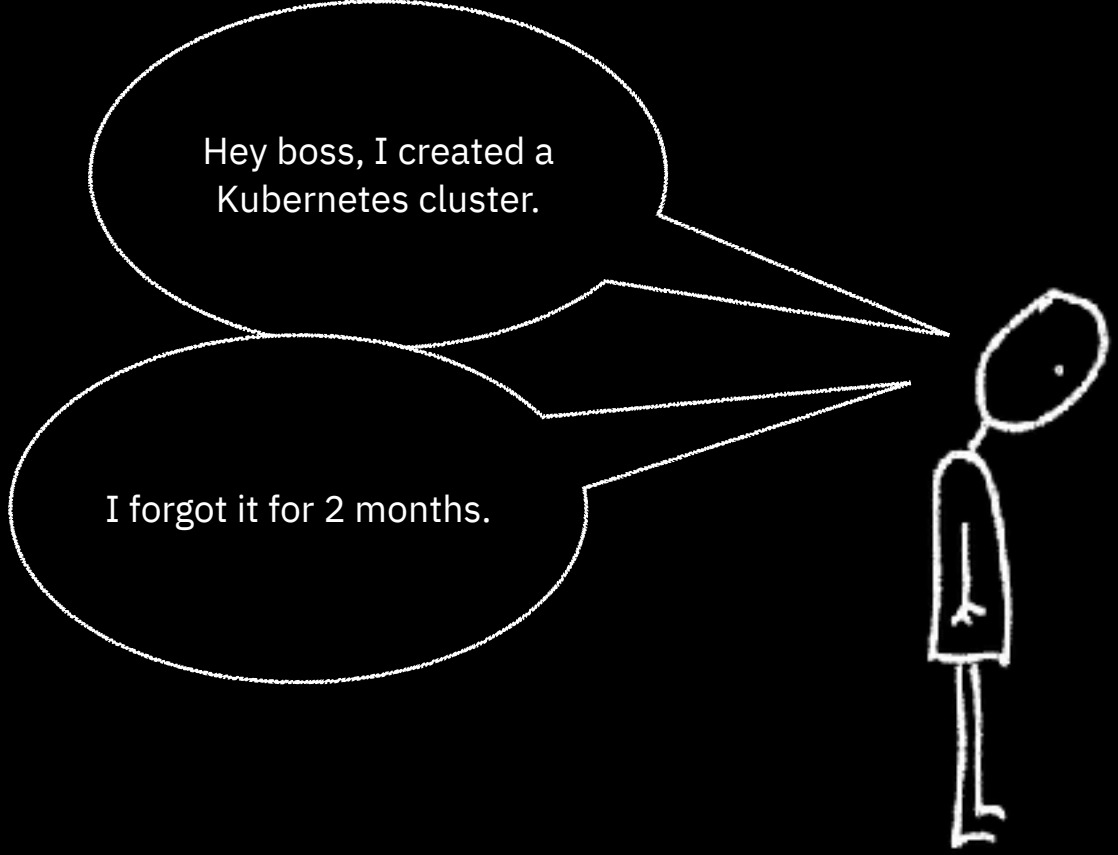
the cloud makes it so **easy**

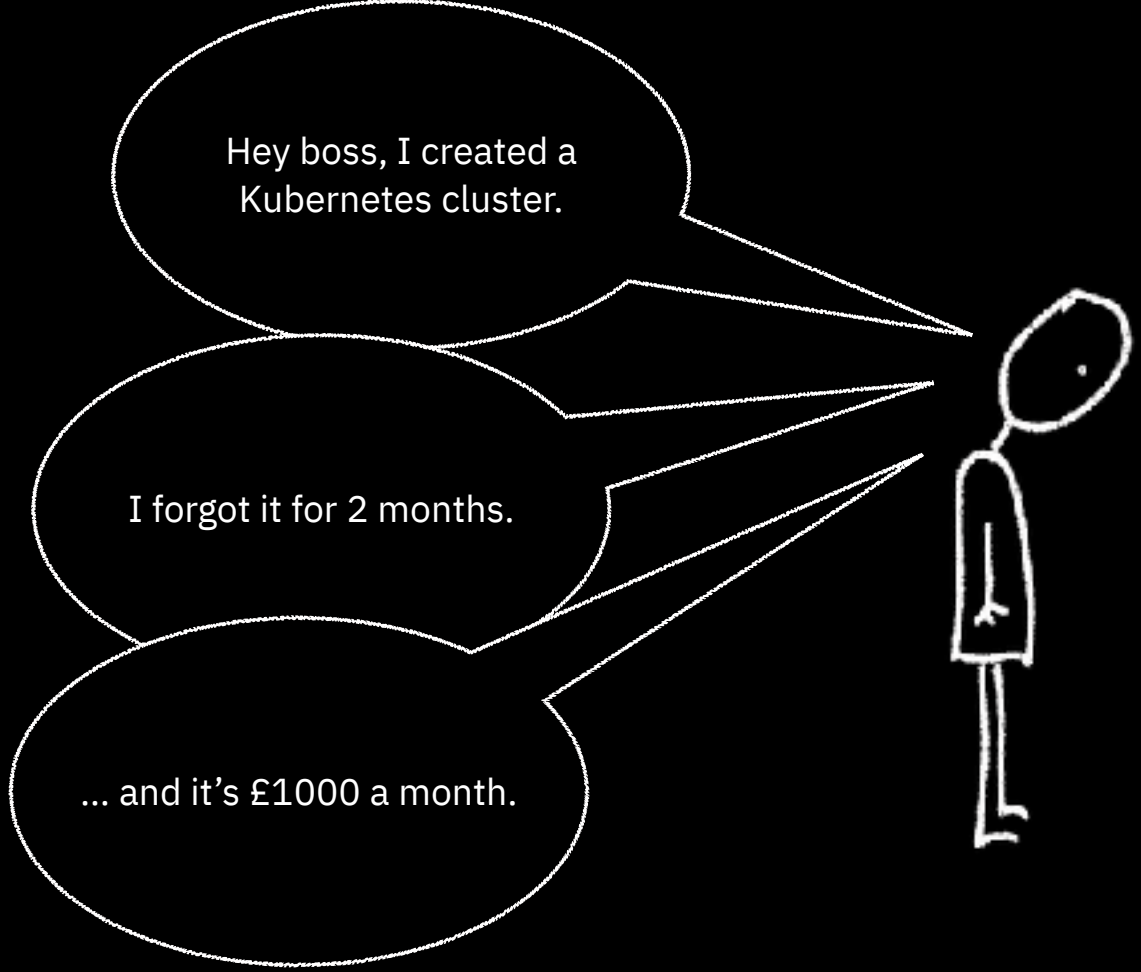
to provision hardware.

that doesn’t mean the hardware is free.

or useful.


Hey boss, I created a Kubernetes cluster.


Hey boss, I created a Kubernetes cluster.
I forgot it for 2 months.


Hey boss, I created a Kubernetes cluster.
I forgot it for 2 months.
... and it’s £1000 a month.

2017 survey

25%

of 16,000 servers
doing no useful work

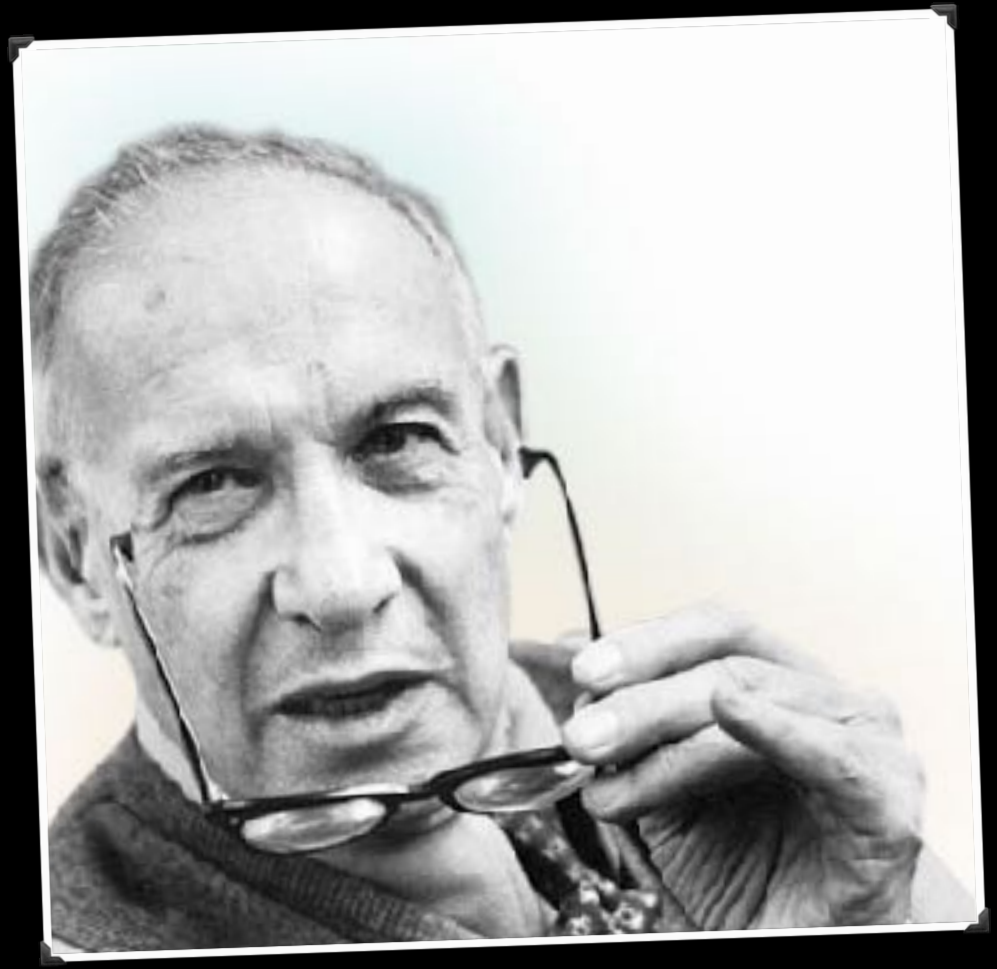
2017 survey

25%

of 16,000 servers
doing no useful work

“perhaps someone forgot to turn them off”


ouch



There is surely nothing quite so useless as doing with great efficiency what should not be done at all.

— Peter Drucker

“we have **no idea** how much we’re spending on cloud.”

cloud to manage your clouds

IBM Multicloud Manager

Create resource | Catalog | Docs | Support

Clusters

Search

Name	Namespace	Labels	Endpoint	Status	Nodes	Klusterlet Version	Kubernetes Version	Storage	Memory	CPU
aks	aksnamespace	cloud=Azure datacenter=eastus environment=Prod name=aks owner=poc region=US vendor=AKS	-	Ready	3	3.1.2-dev	v1.12.6	-	3%	9%
eks	eksnamespace	cloud=AWS datacenter=us-east-1 environment=public name=eks owner=poc region=US vendor=EKS	-	Ready	3	3.1.2-dev	v1.11.8-eks-7c34c0	-	5%	12%
gke	gkenamespace	cloud=Google datacenter=us-east1-b environment=Prod name=gke owner=poc region=US vendor=GKE	-	Ready	3	3.1.2-dev	v1.11.7-gke.12	-	8%	35%
icp	icpnamespace	cloud=IBM datacenter=frankfurt environment=Dev name=icp owner=poc region=EU vendor=ICP	launch	Ready	5	3.1.2-dev	v1.12.4+icp-ee	100%	35%	17%
iks	iksnamespace	cloud=IBM datacenter=berlin environment=secure location=public name=iks owner=poc region=EU vendor=IKS	-	Ready	2	3.1.2-dev	v1.11.3+IKS	-	59%	44%
		cloud=IBM datacenter=frankfurt								

FinOps

fail

microservices ops mayhem

SRE

site reliability engineering

@notty_cummins

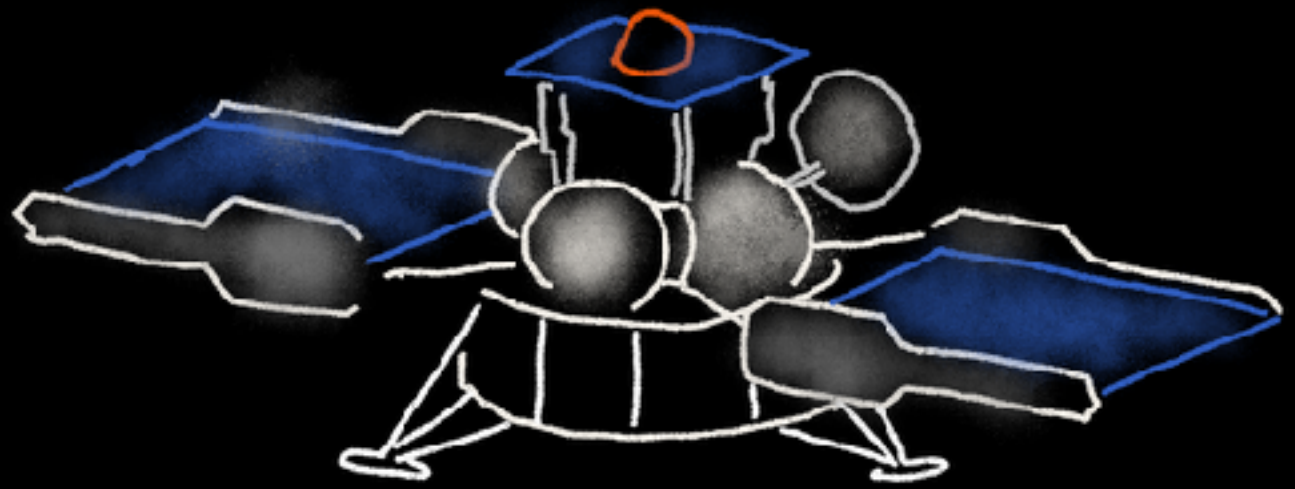
make

releases

deeply **boring**

@notty_cummins

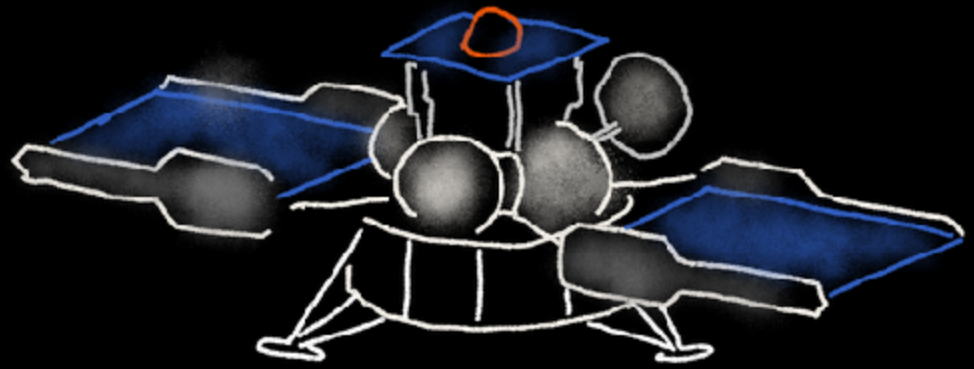
how to brick a spaceprobe



Phobos 1

“we couldn’t get the automated checks to work, so we bypassed them”

recoverability



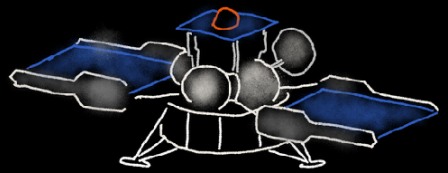
unrecoverable

back in ms
no data loss

bricked

manual
intervention

fast, but
data lost



handoffs

handoffs bad
automation good

ways to succeed at cloud native

be clear on what you're trying to achieve

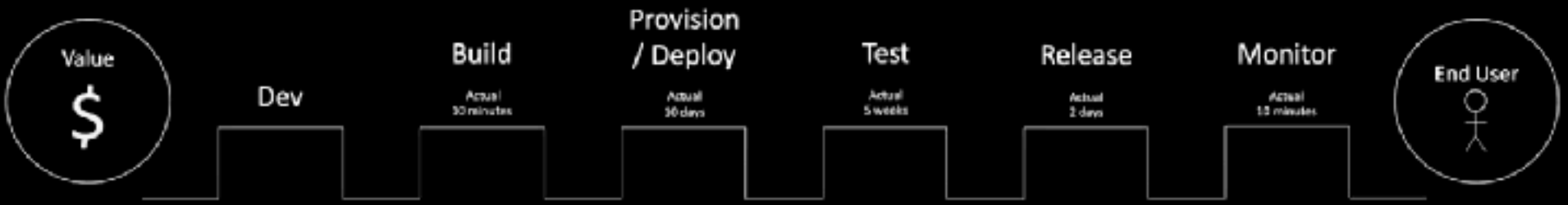
optimise for feedback

if you automate something, change the processes around that assume that the previously manual process is expensive or error prone.

Delivering software better

The objective is to...

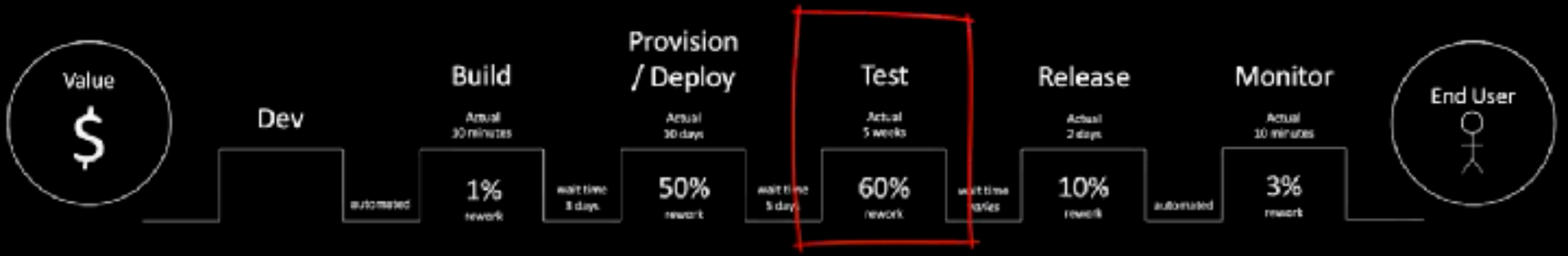
Optimize the System as a Whole



Value → User

Delivering software better

Localized optimization will not deliver desired outcomes!



@holly_cummins