Nine Ways To **Fail** At Cloud Native

Holly Cummins
IBM **Garage**
@holly_cummins

An expert is a person who has found out by their own painful experience all the mistakes that one can make in a very narrow field.

— Niels Bohr

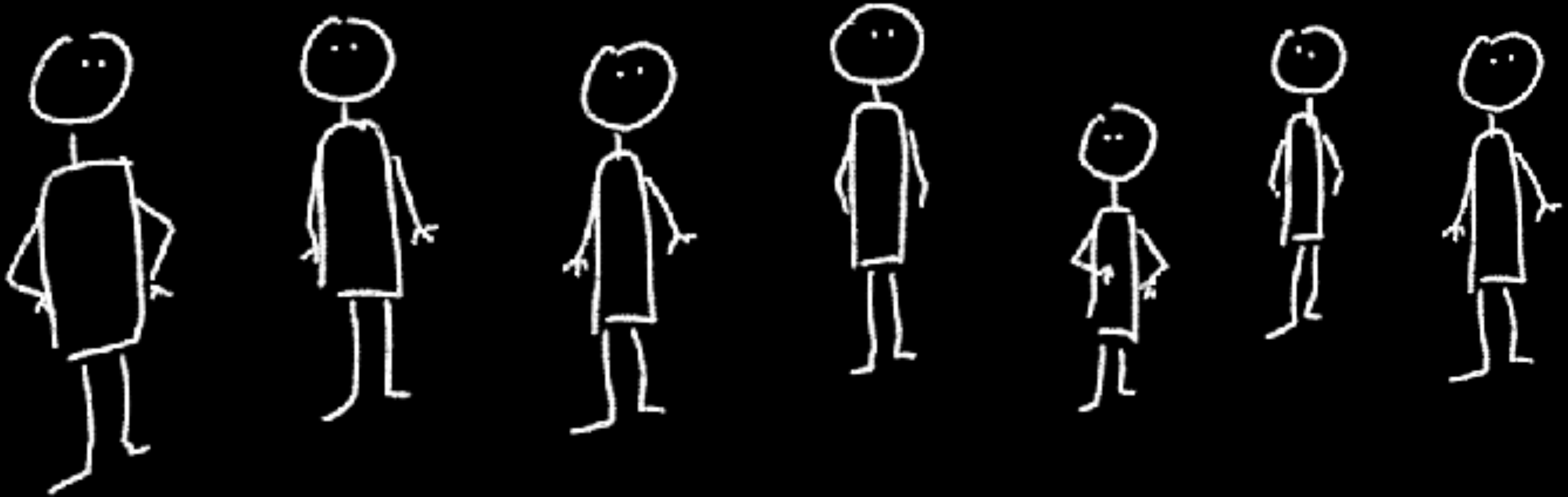
I’m a consultant with the IBM Garage.

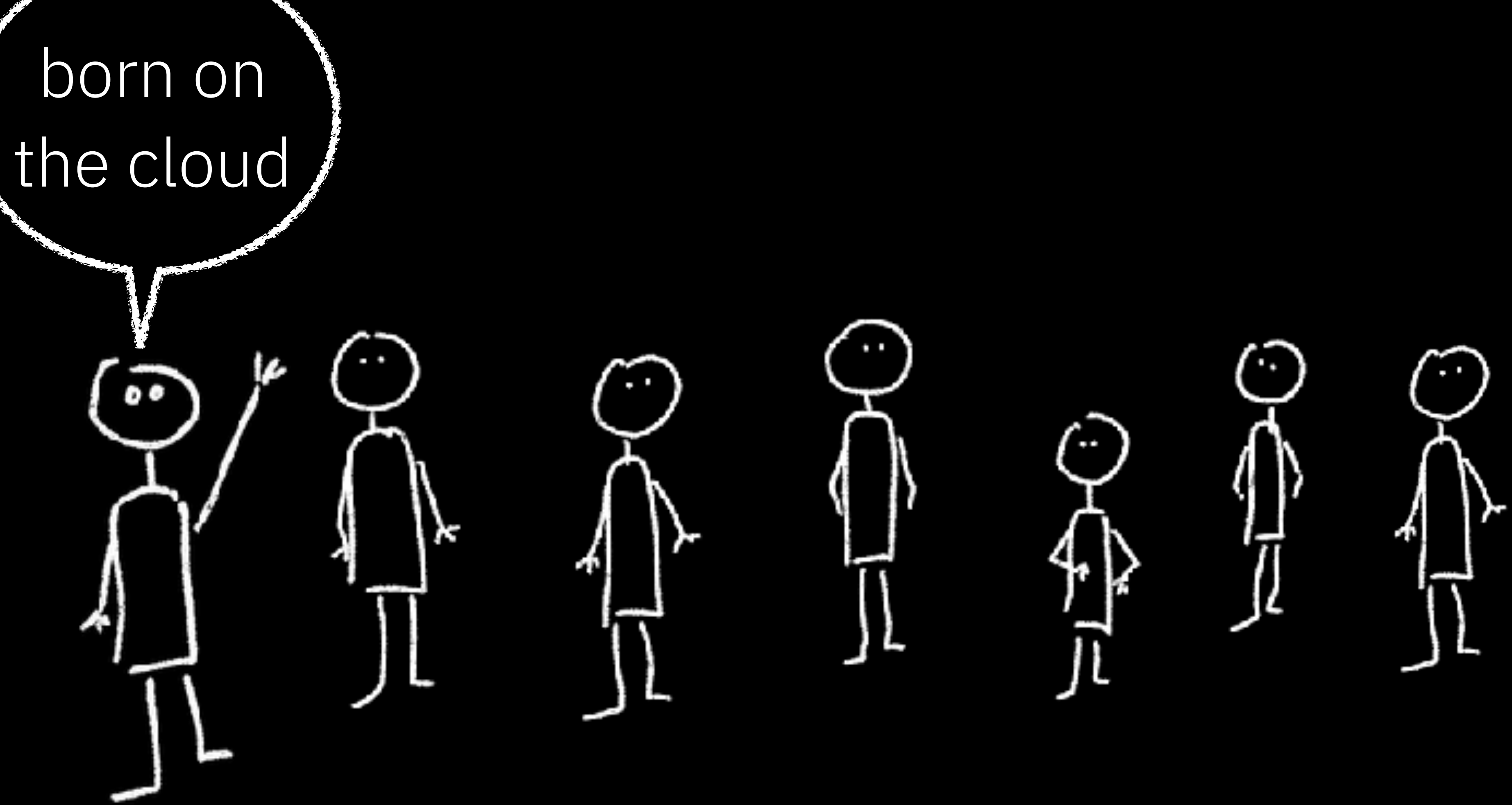
These are my scary stories

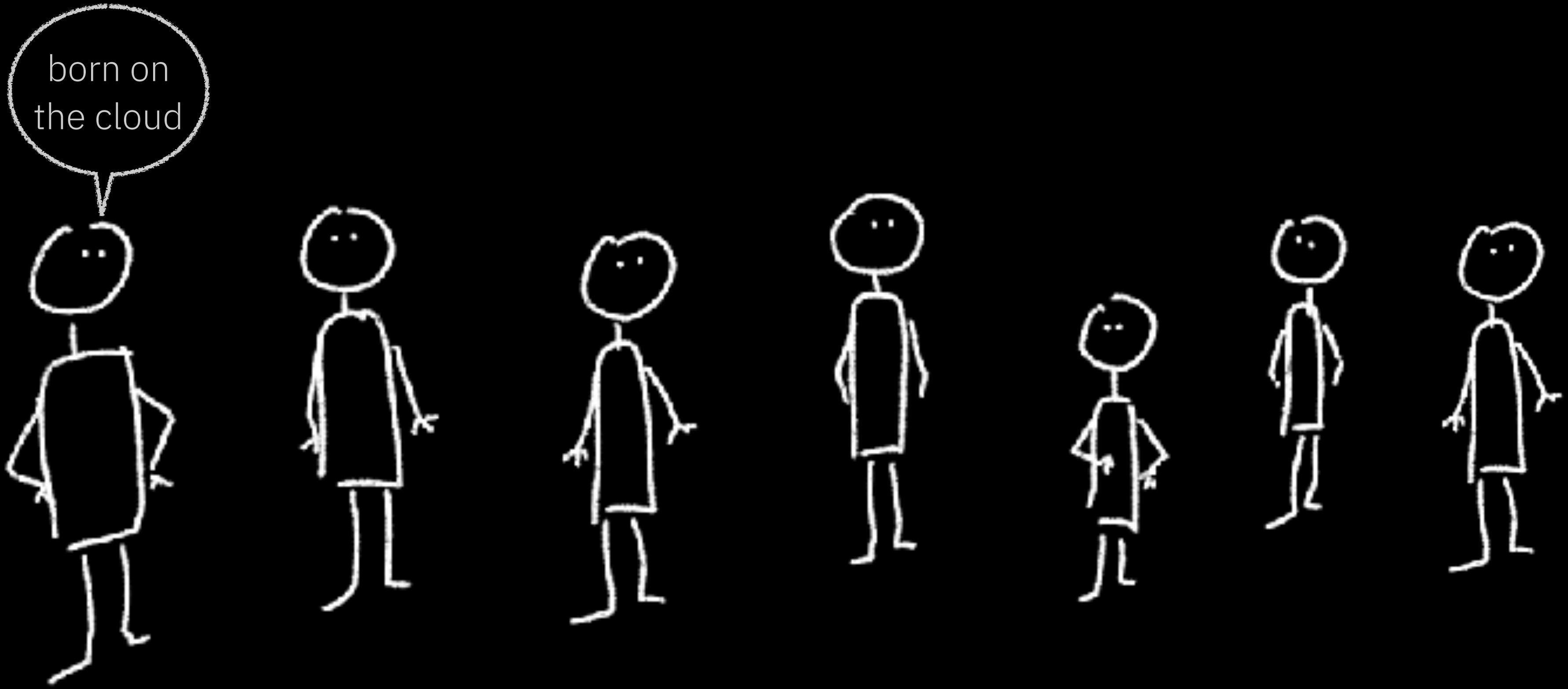
fail 1

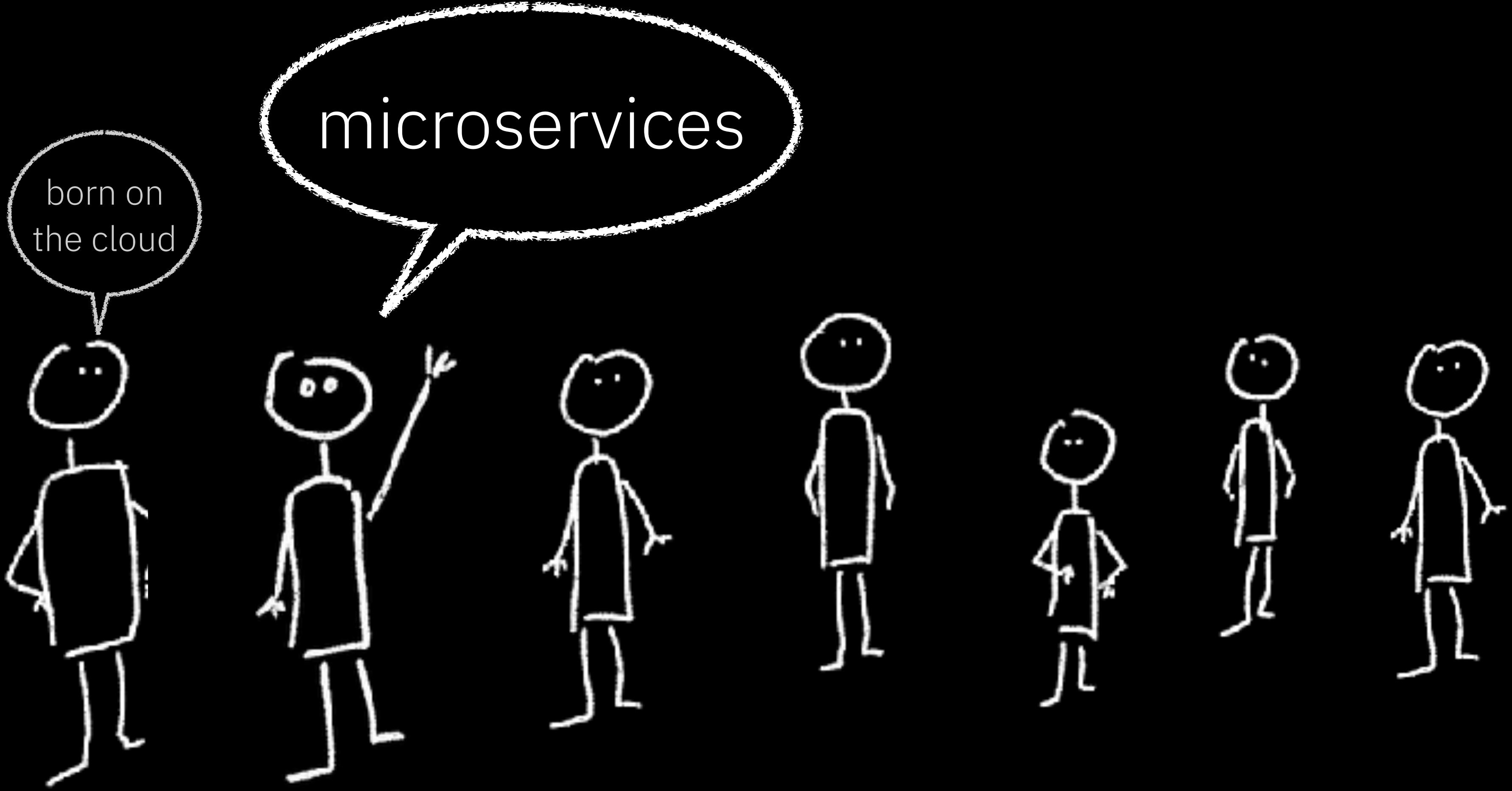
the magic morphing meaning

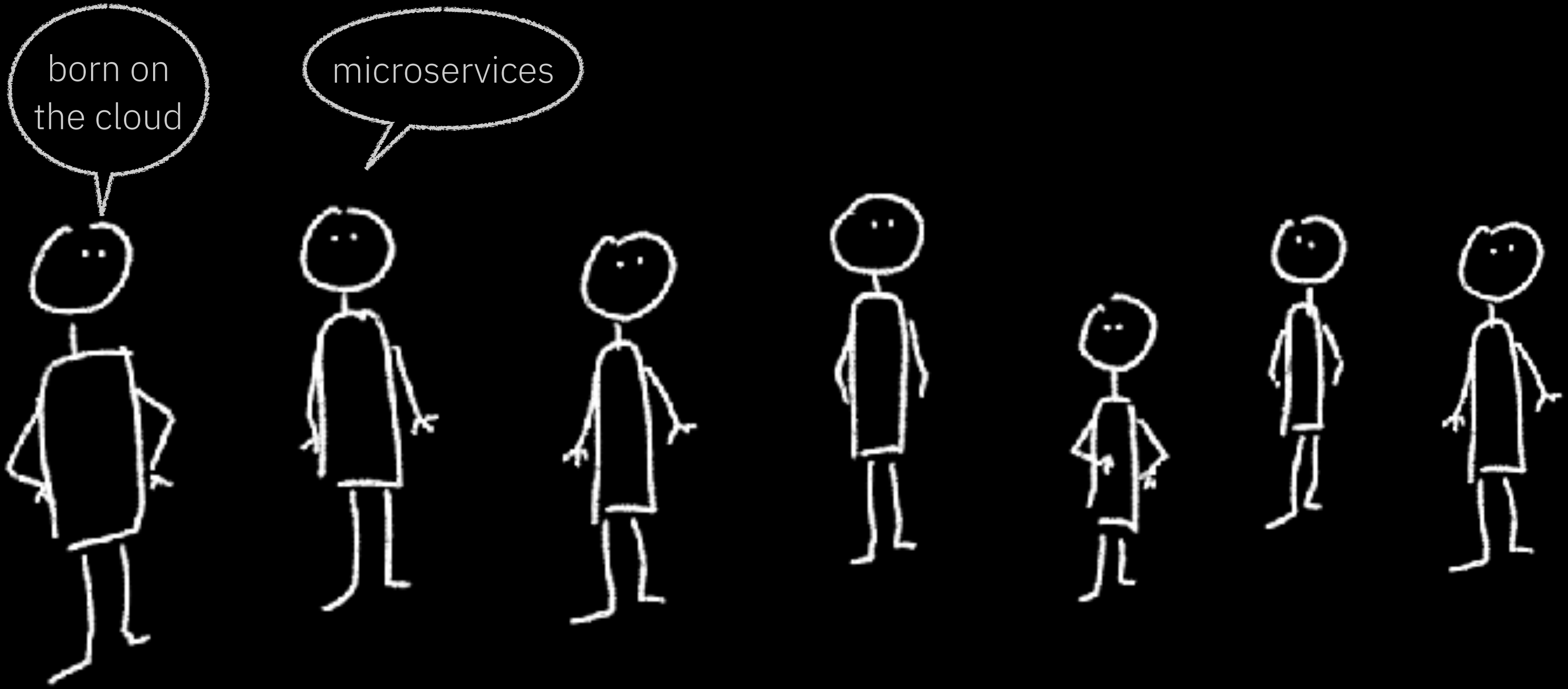
so, what **is** cloud native?

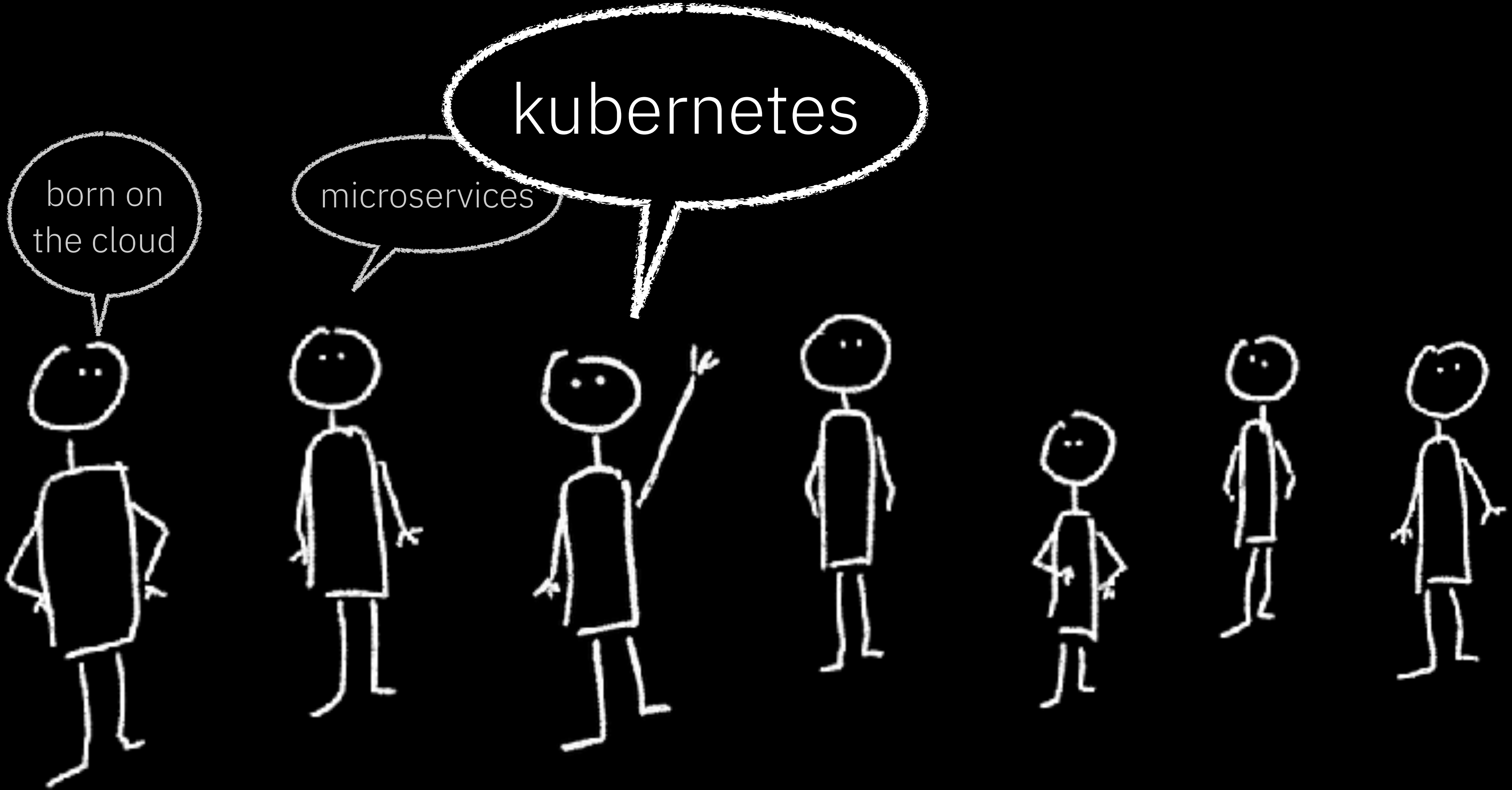


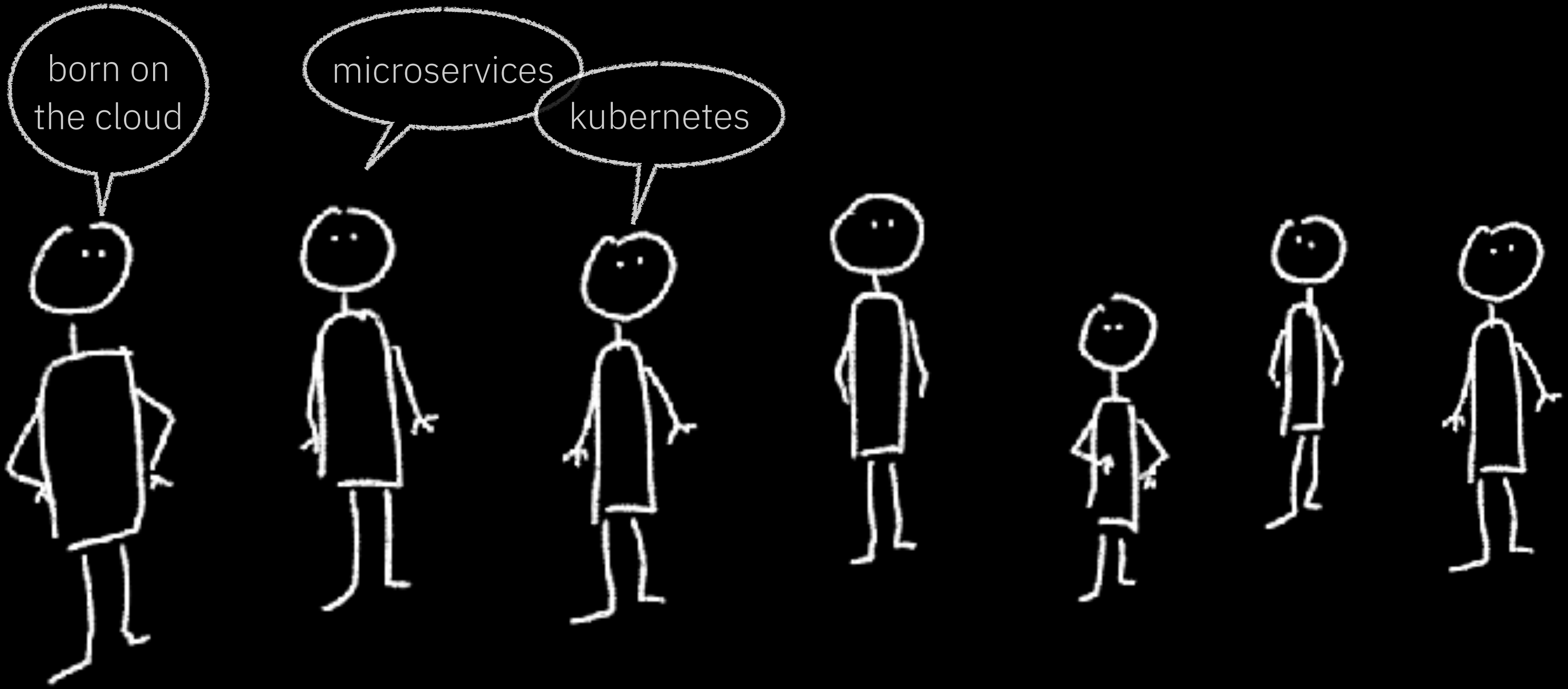

born on the cloud

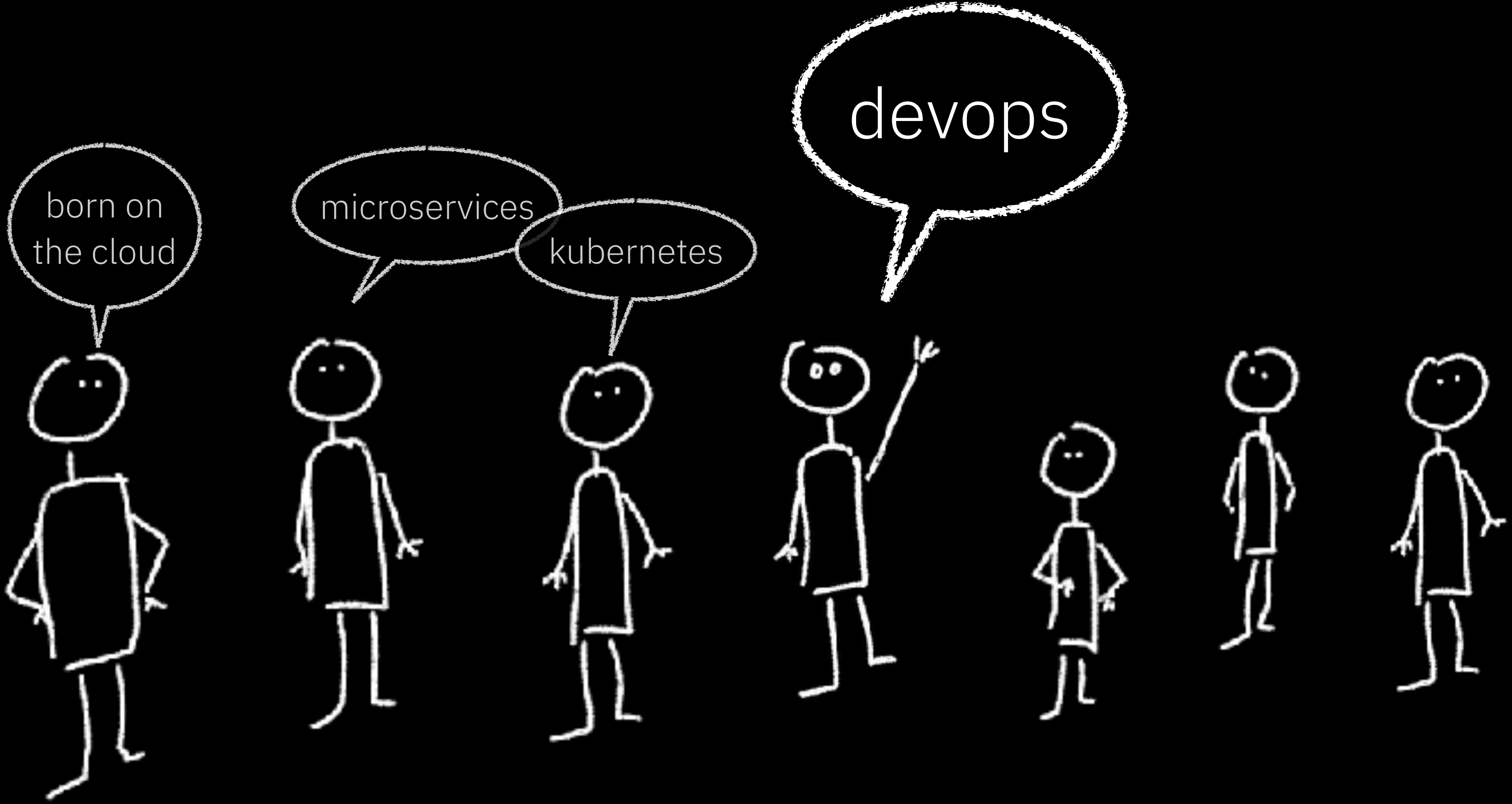

born on the cloud

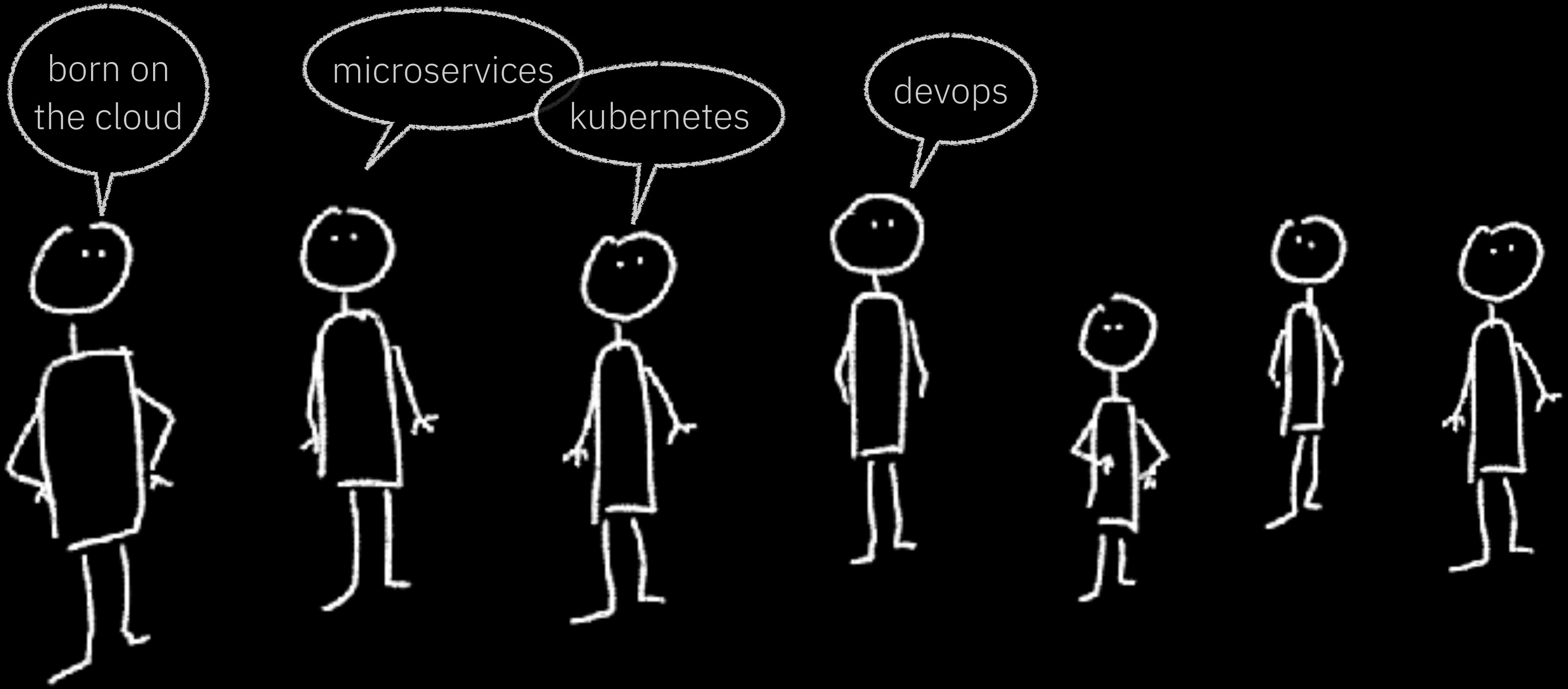

born on the cloud
microservices

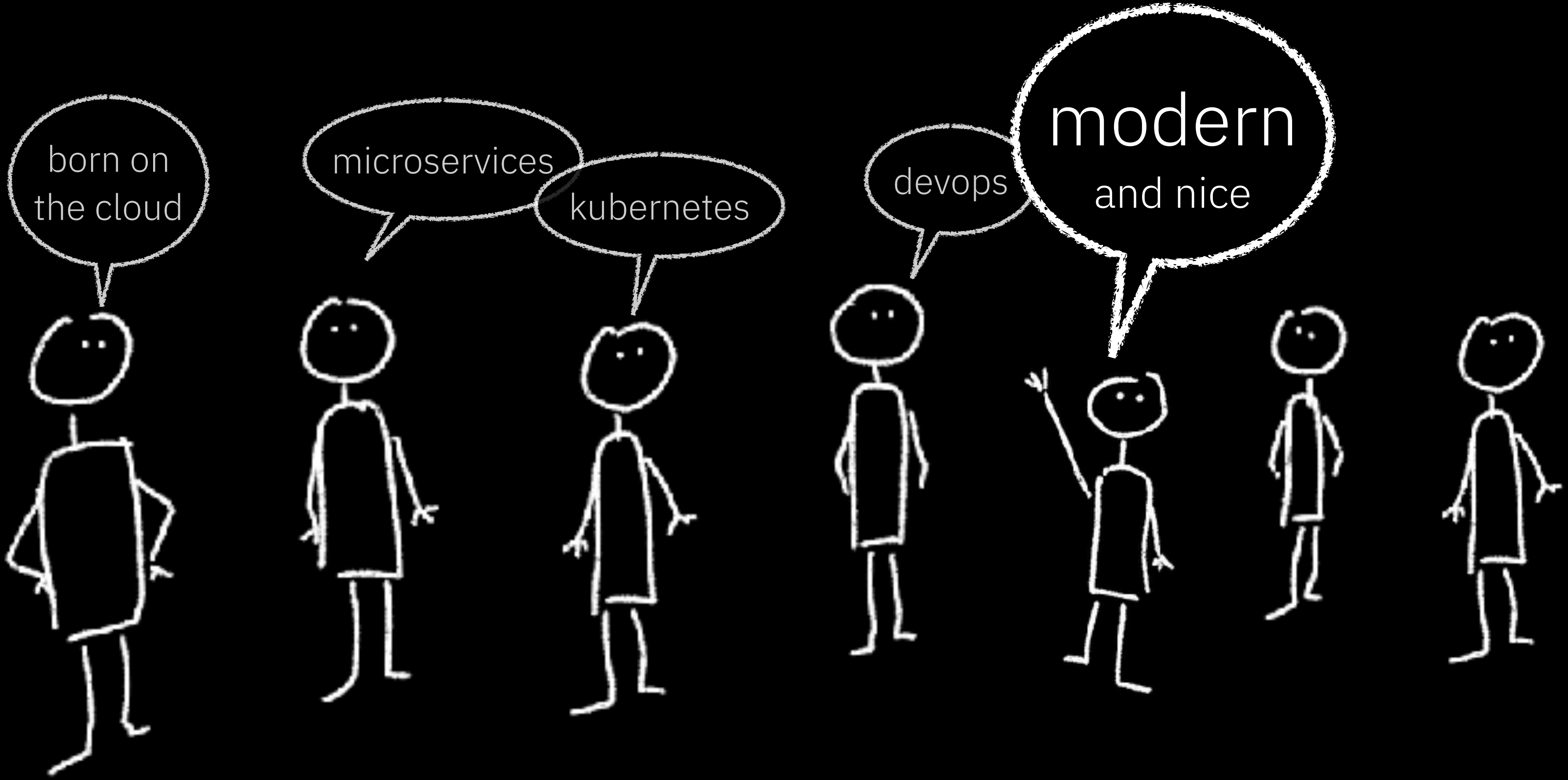

born on the cloud
microservices

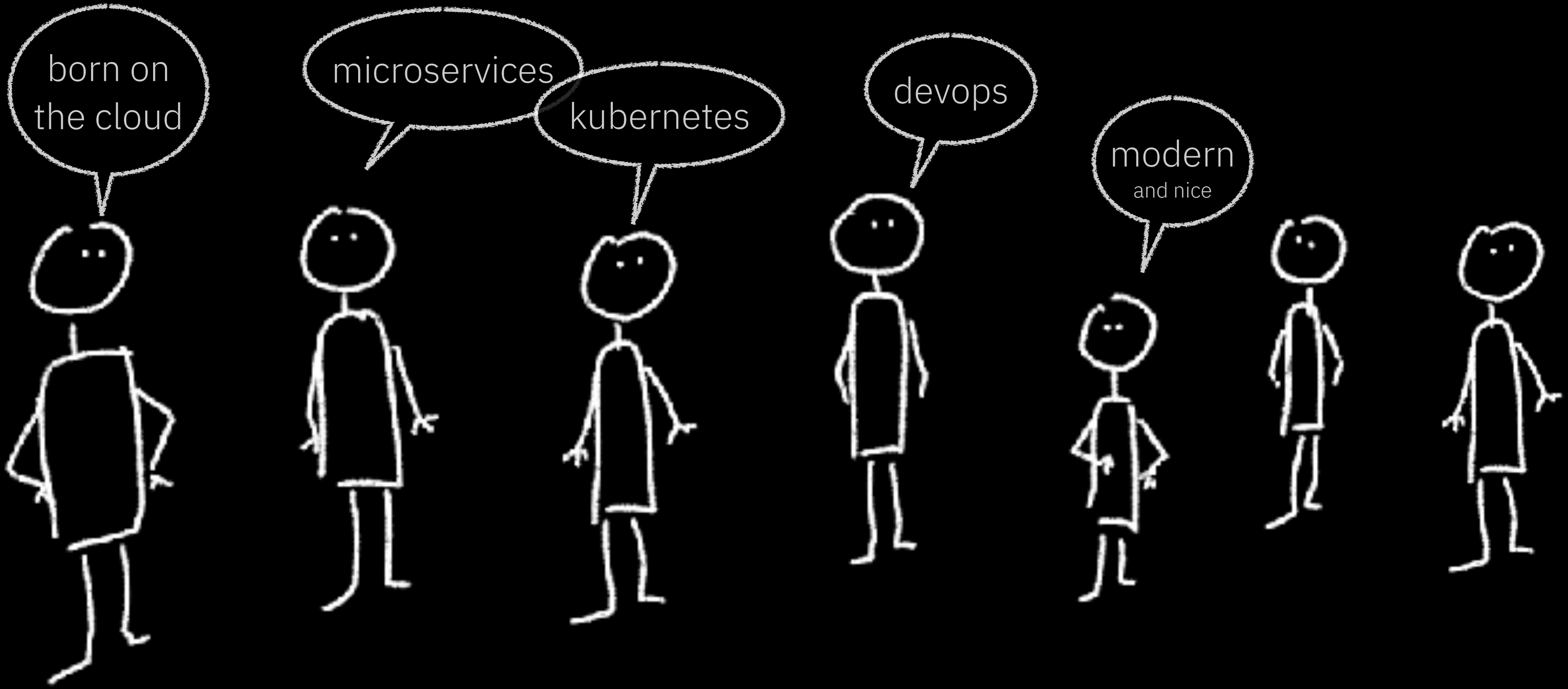

born on the cloud
microservices
kubernetes

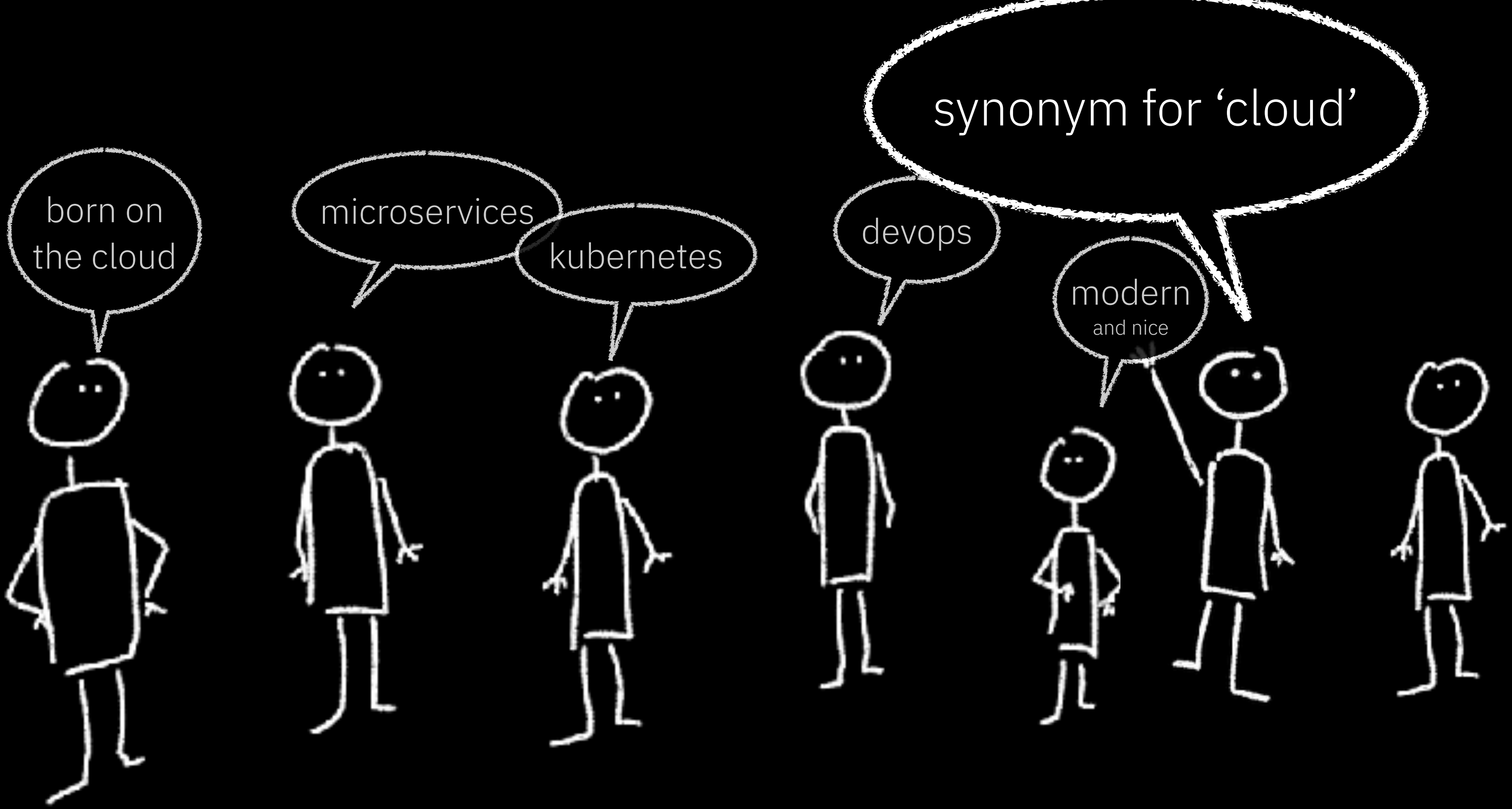

born on the cloud
microservices
kubernetes

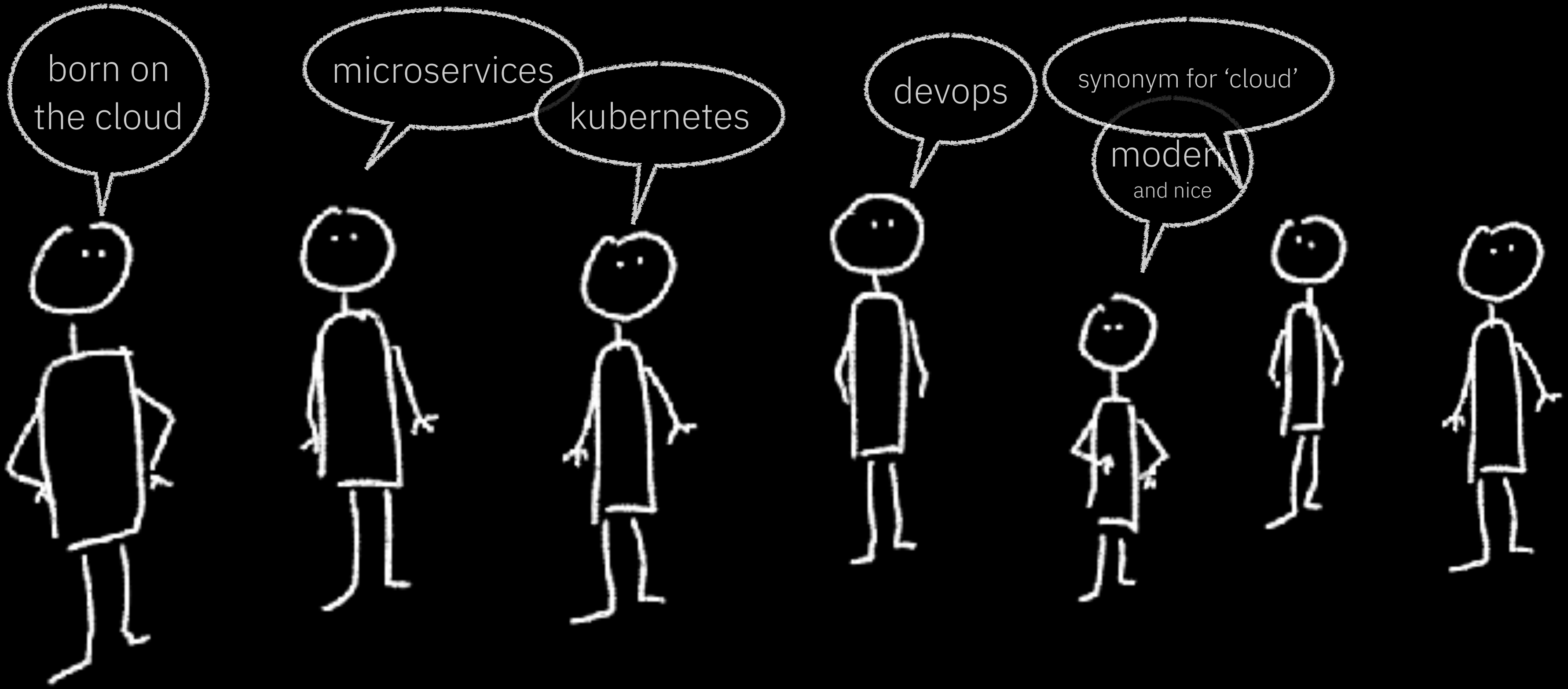

born on the cloud
microservices
kubernetes
devops

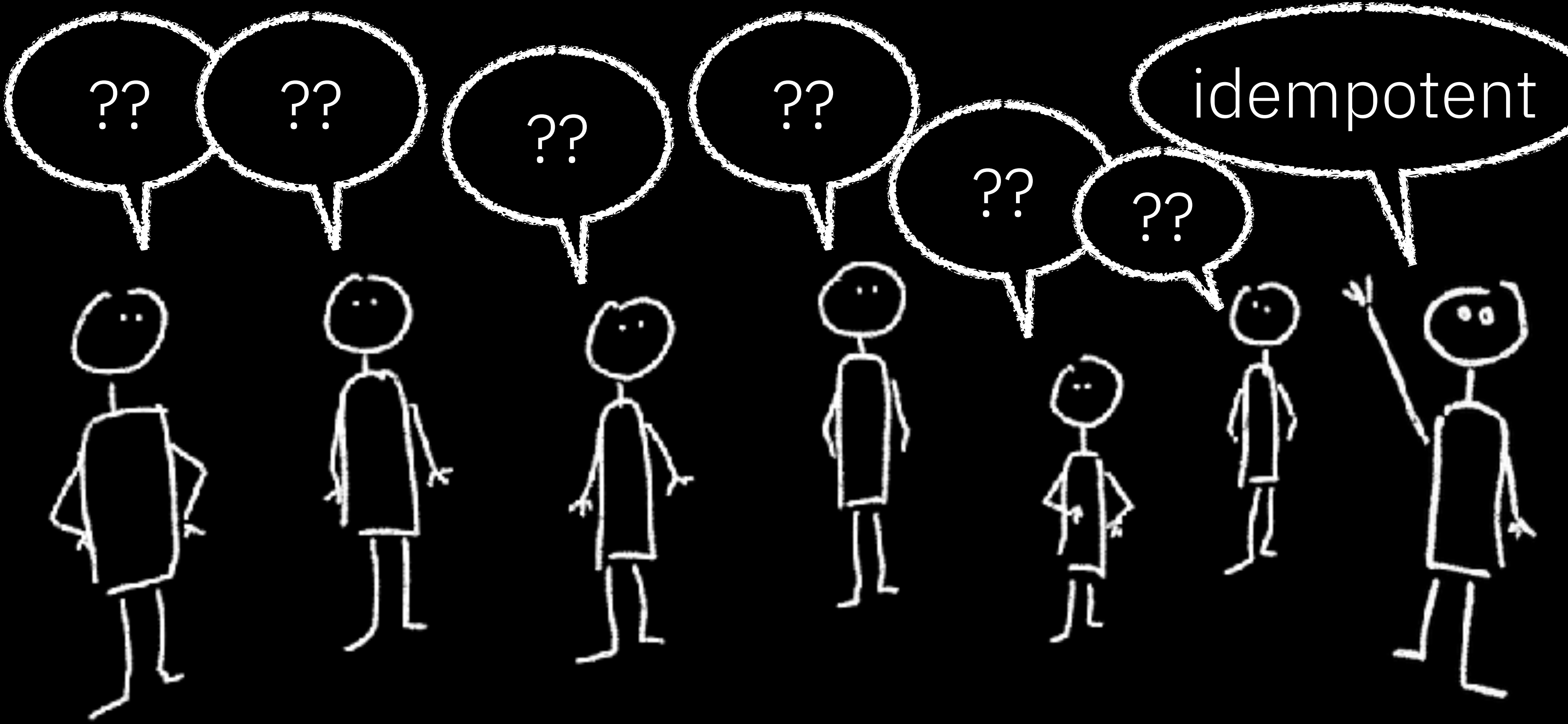

born on the cloud
microservices
kubernetes
devops

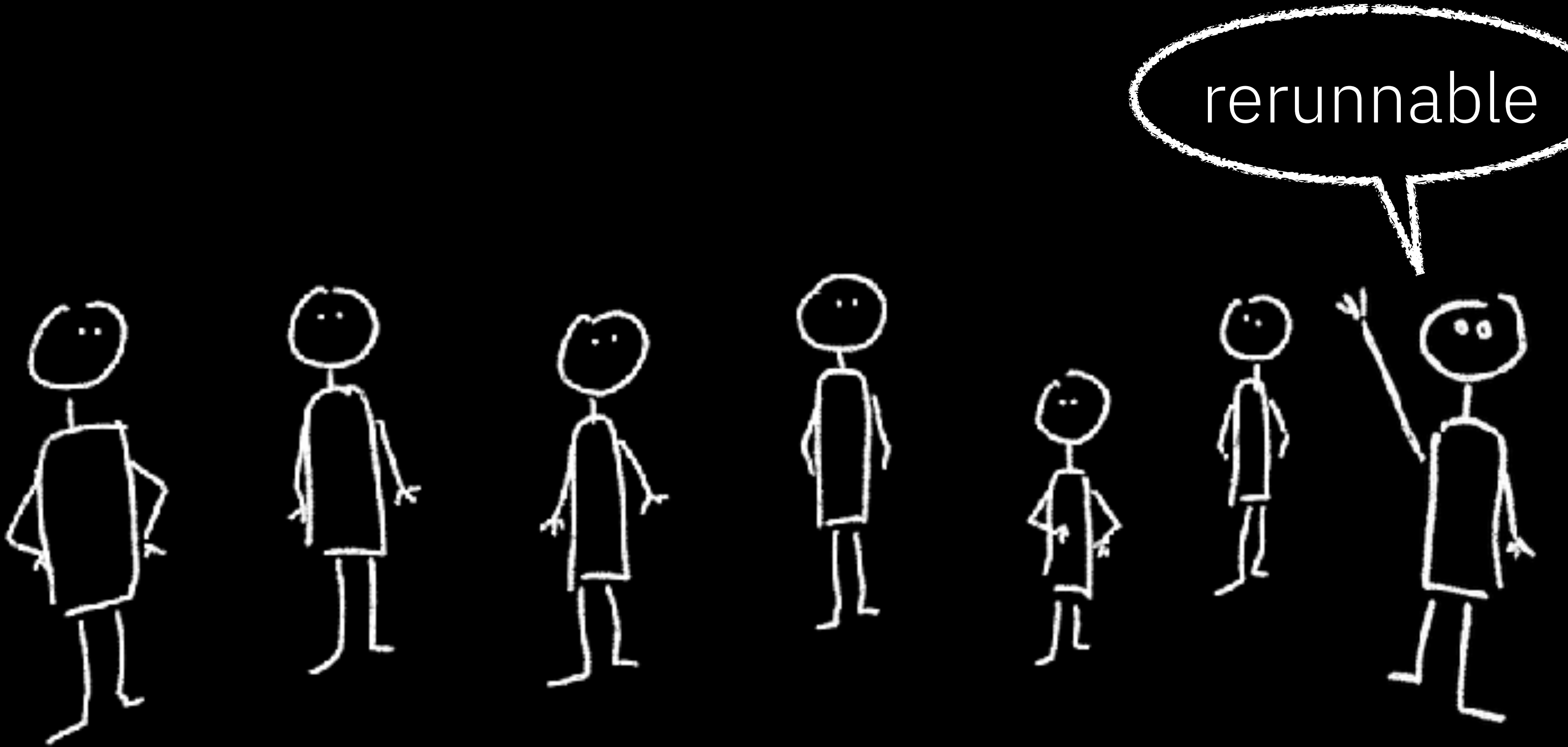

born on the cloud
microservices
kubernetes
devops
modern and nice

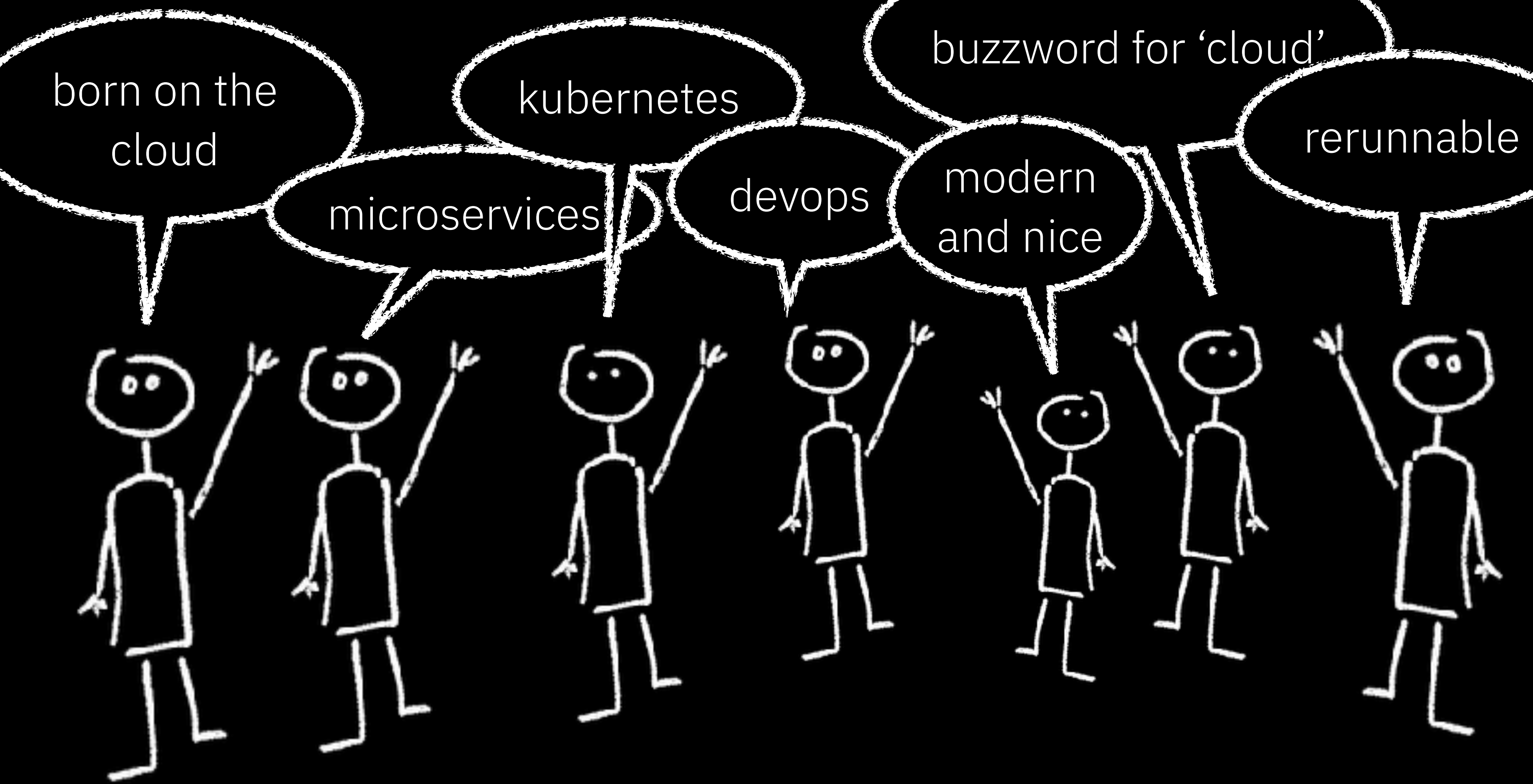

born on the cloud
microservices
kubernetes
devops
modern
and nice


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


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


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


rerunnable


born on the cloud
microservices
kubernetes
devops
modern and nice
buzzword for 'cloud'
rerunnable

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

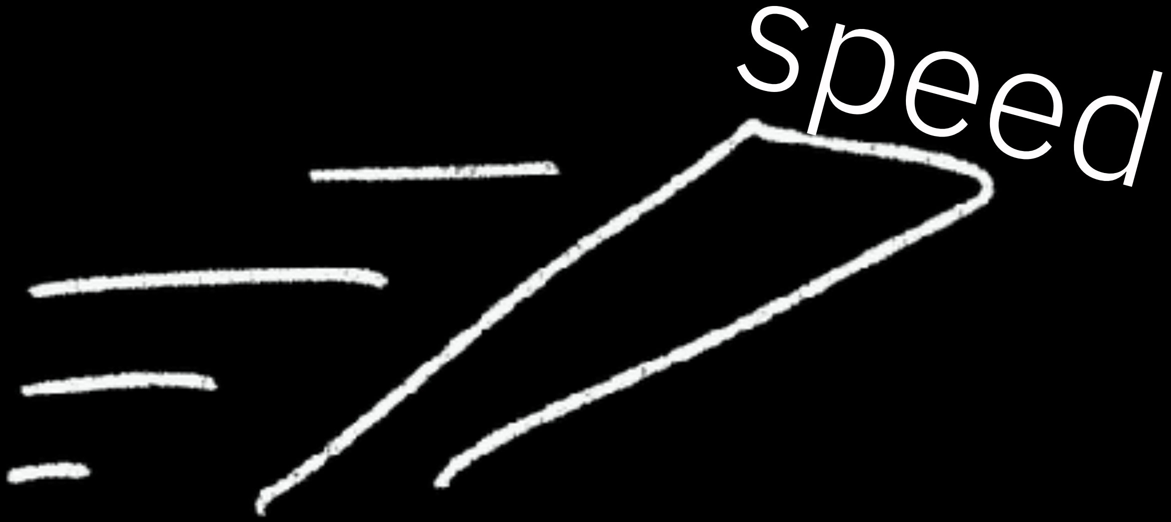
fail 2

the muddy goal

why cloud?

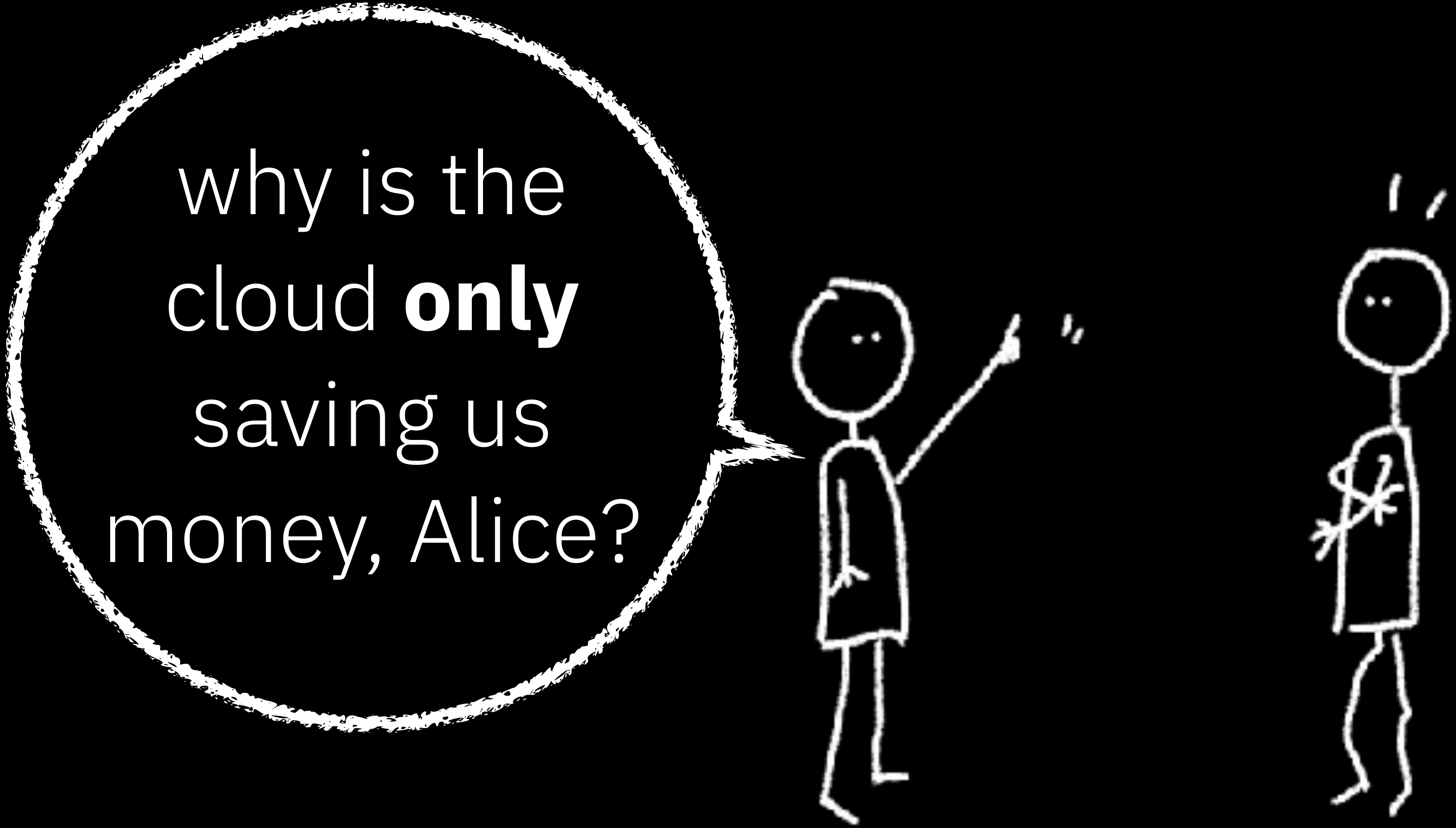
cost

e l a s t i c i t y


speed

exotic capabilities

security


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

fail 3

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 …

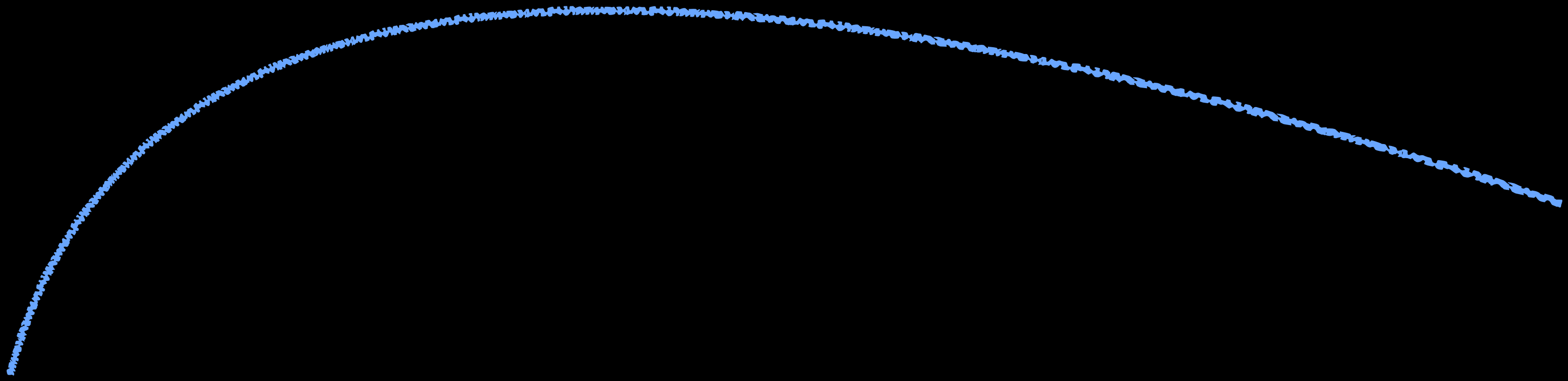
we release every six months …

CI/CD …. ”

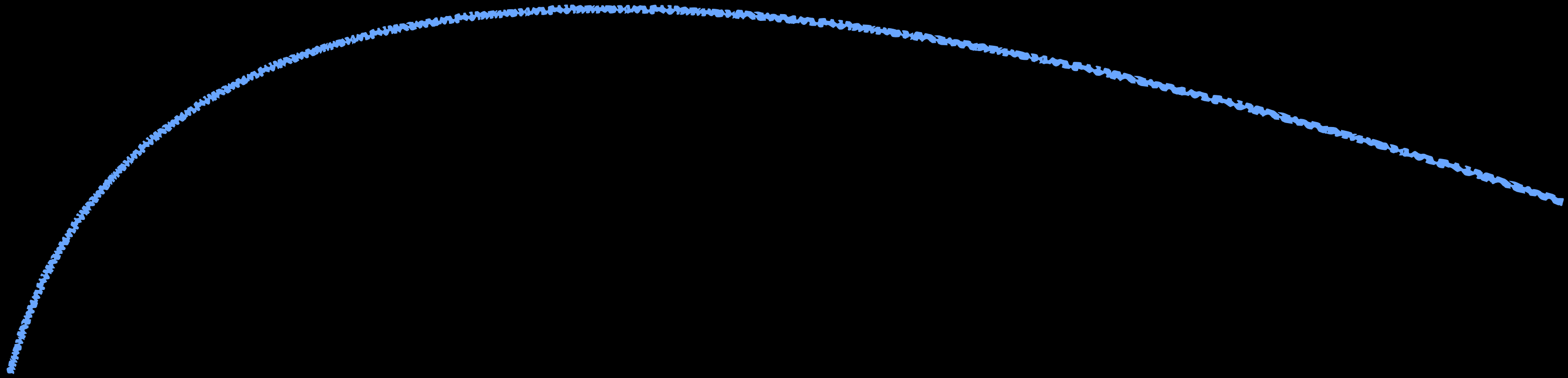
continuous.

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

how often should you push to master?

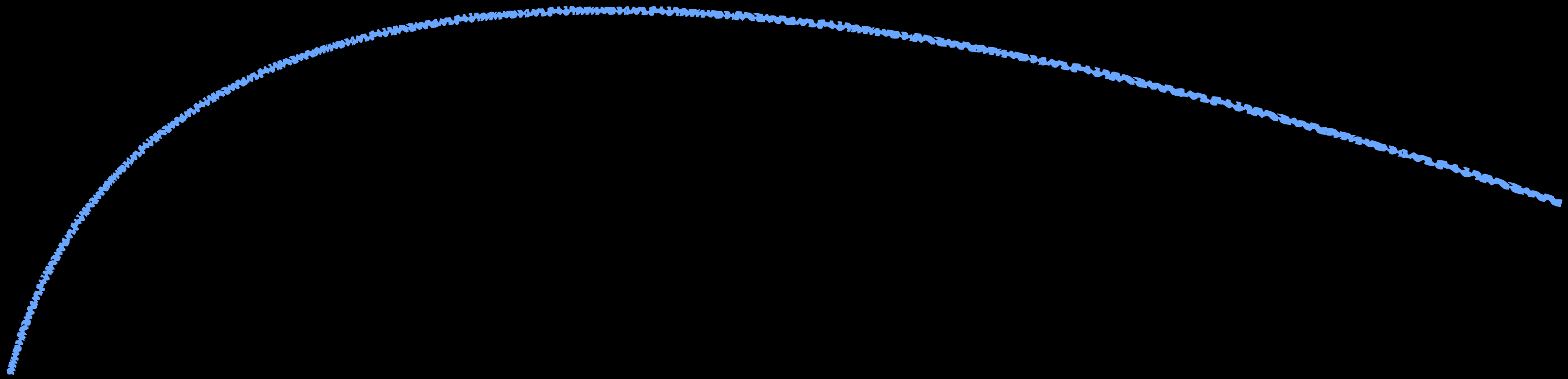


how often should you integrate?



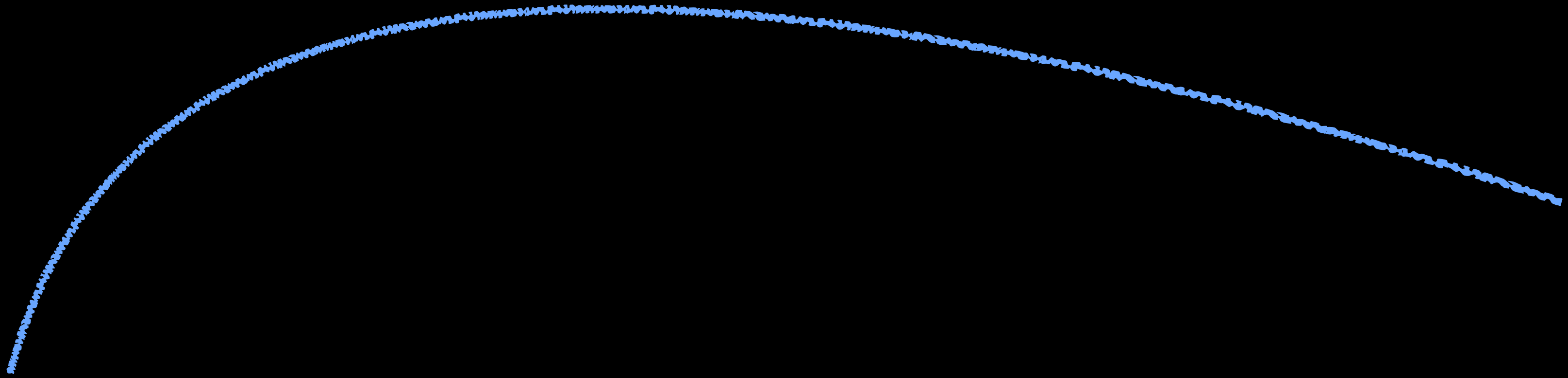
how often should you integrate?

every character



how often should you integrate?

every character



actually continuous
... but stupid

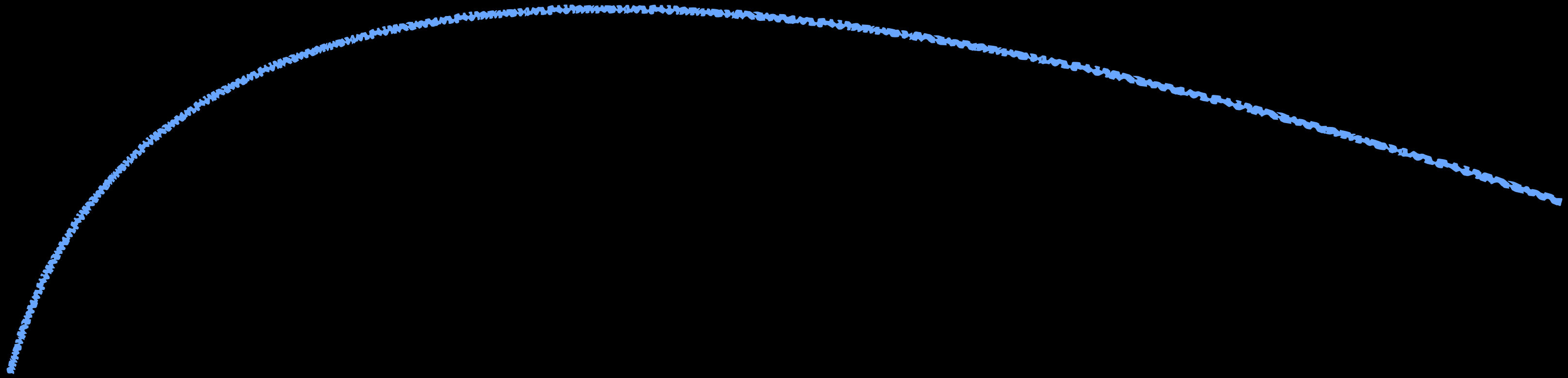
#IBMGarage

@holly_cummins

how often should you integrate?

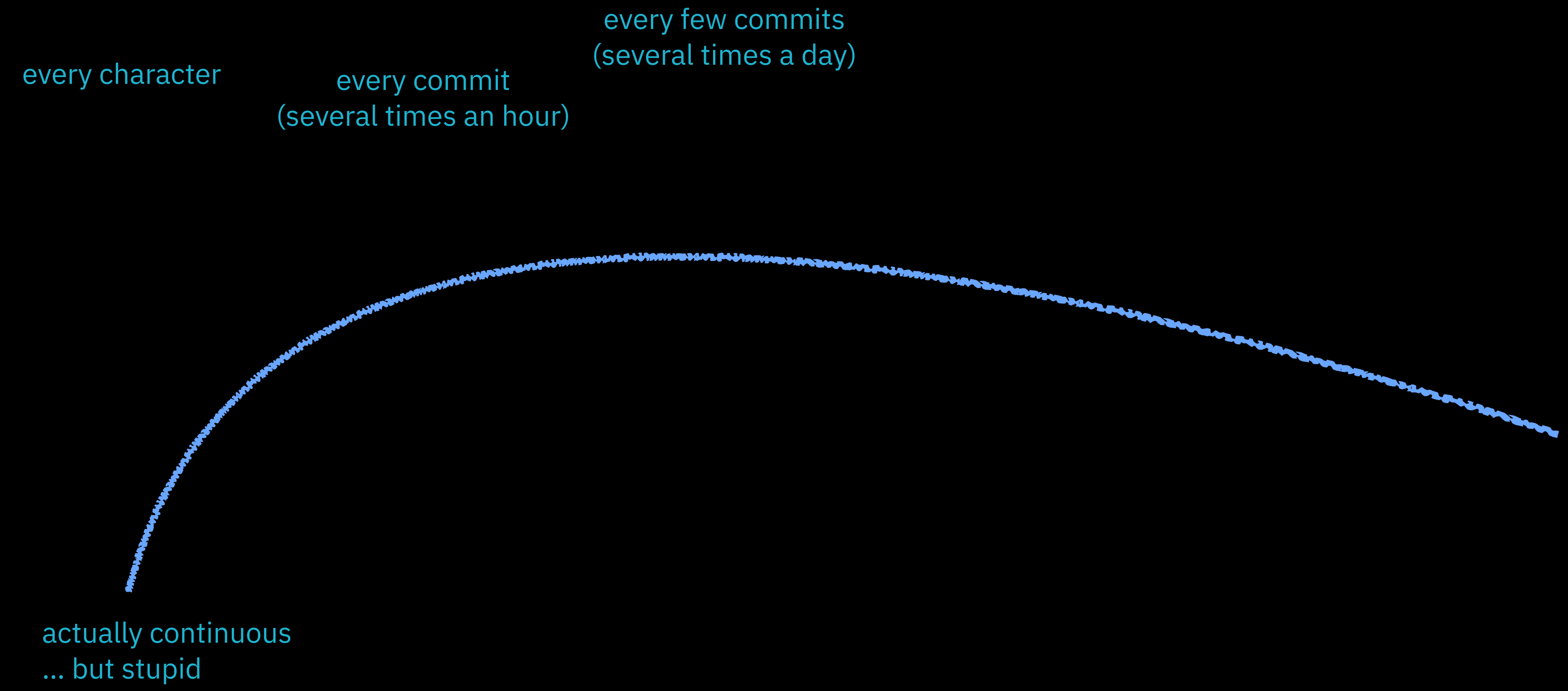
every character

every commit
(several times an hour)

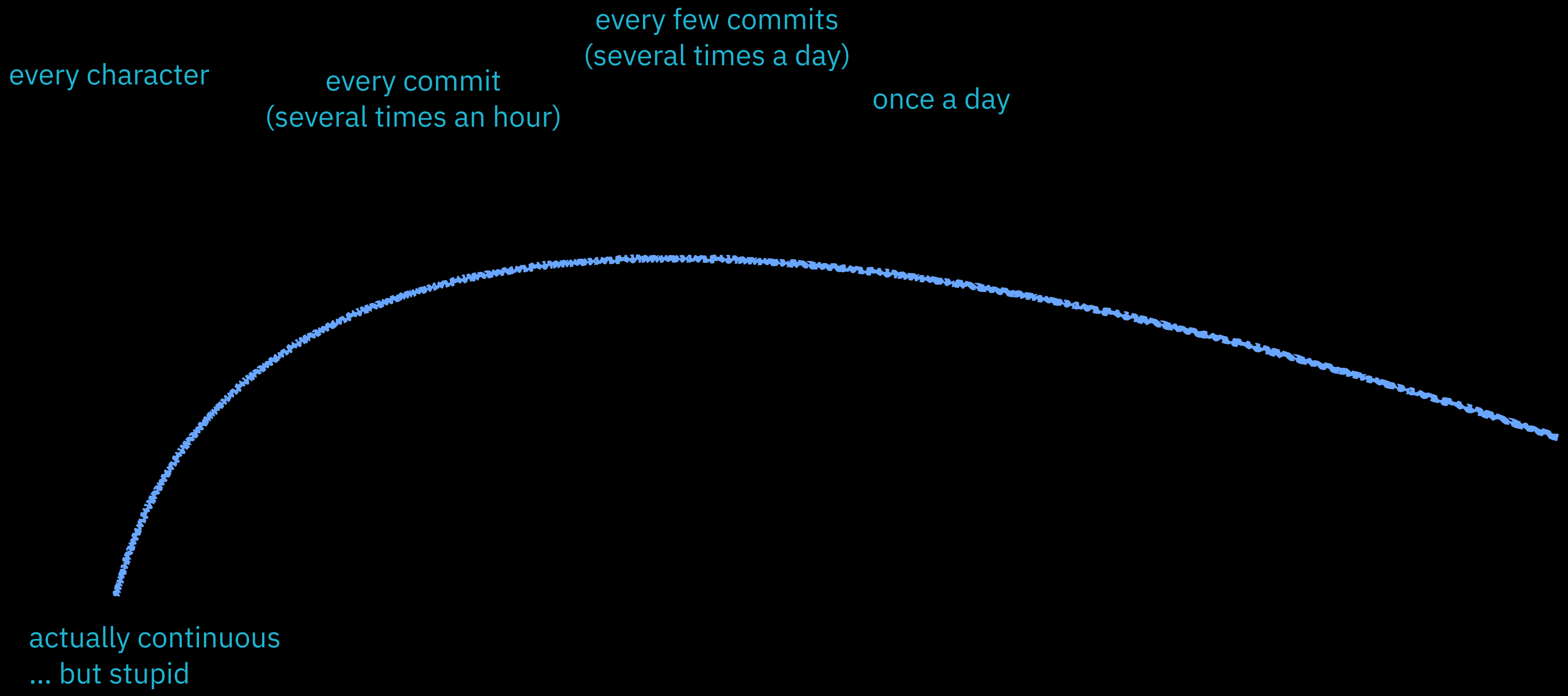


actually continuous
… but stupid

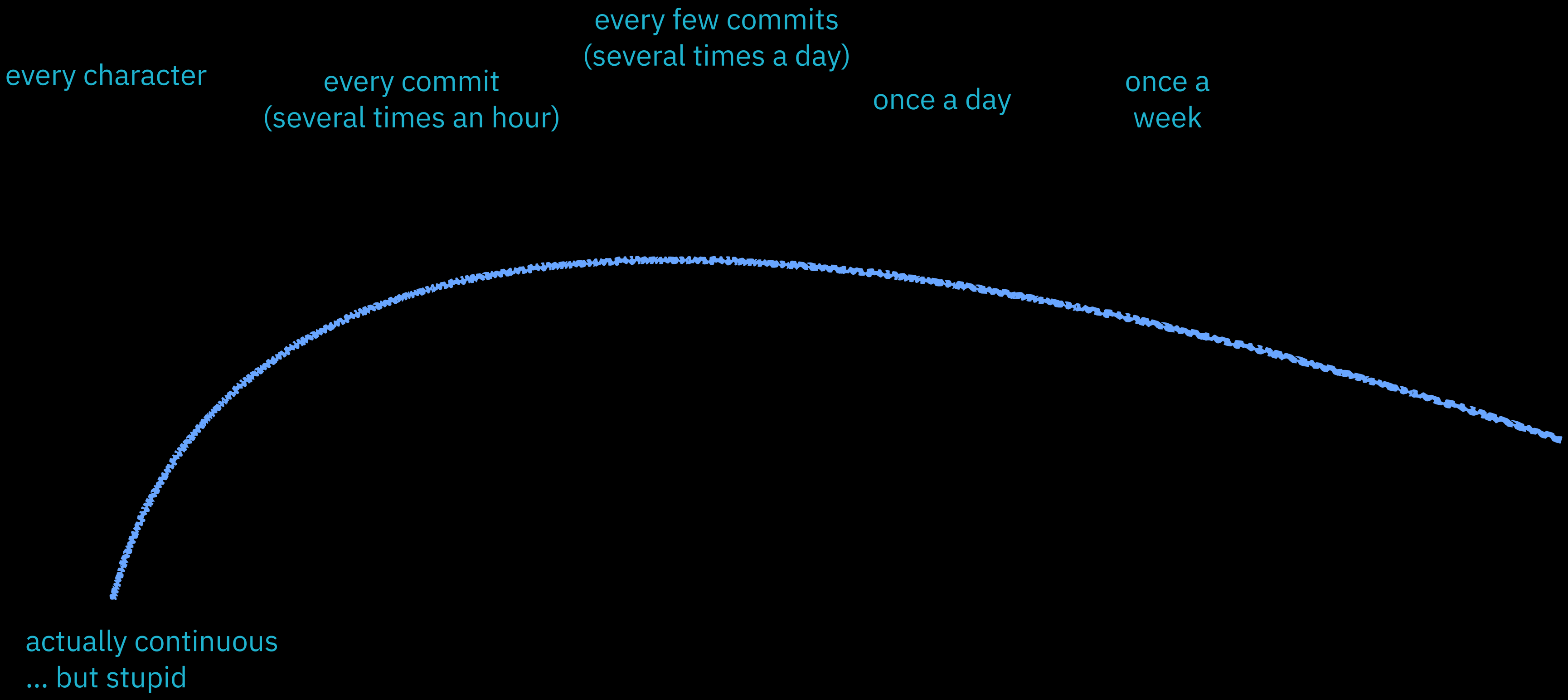
how often should you integrate?



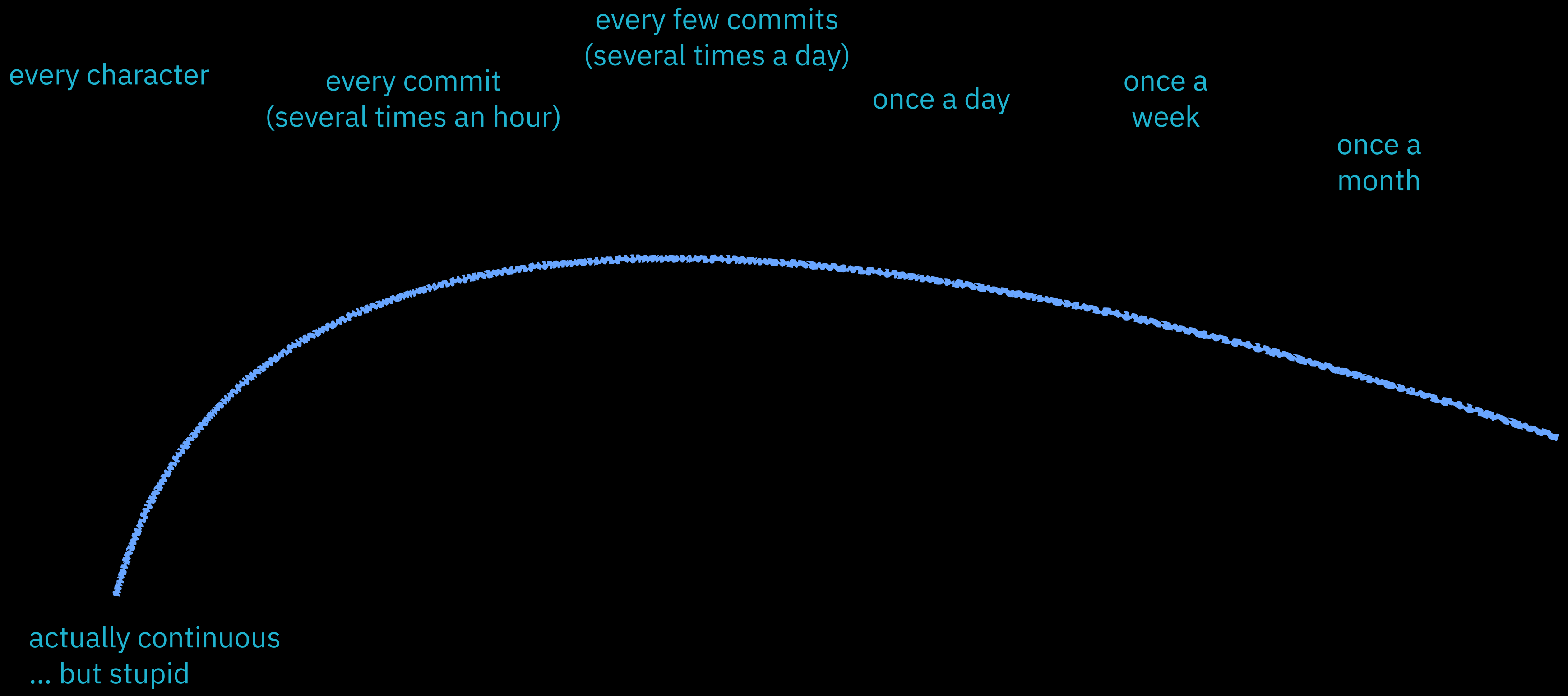
how often should you integrate?



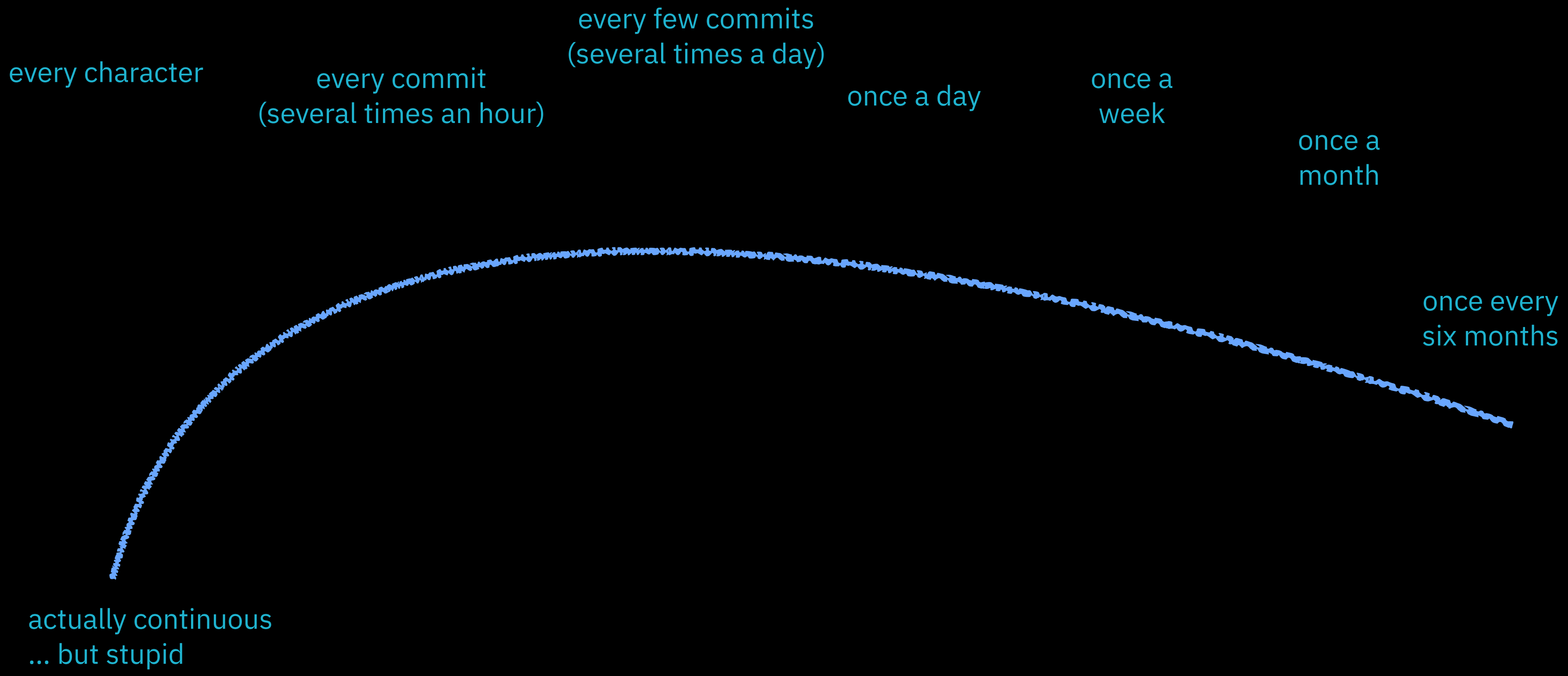
how often should you integrate?



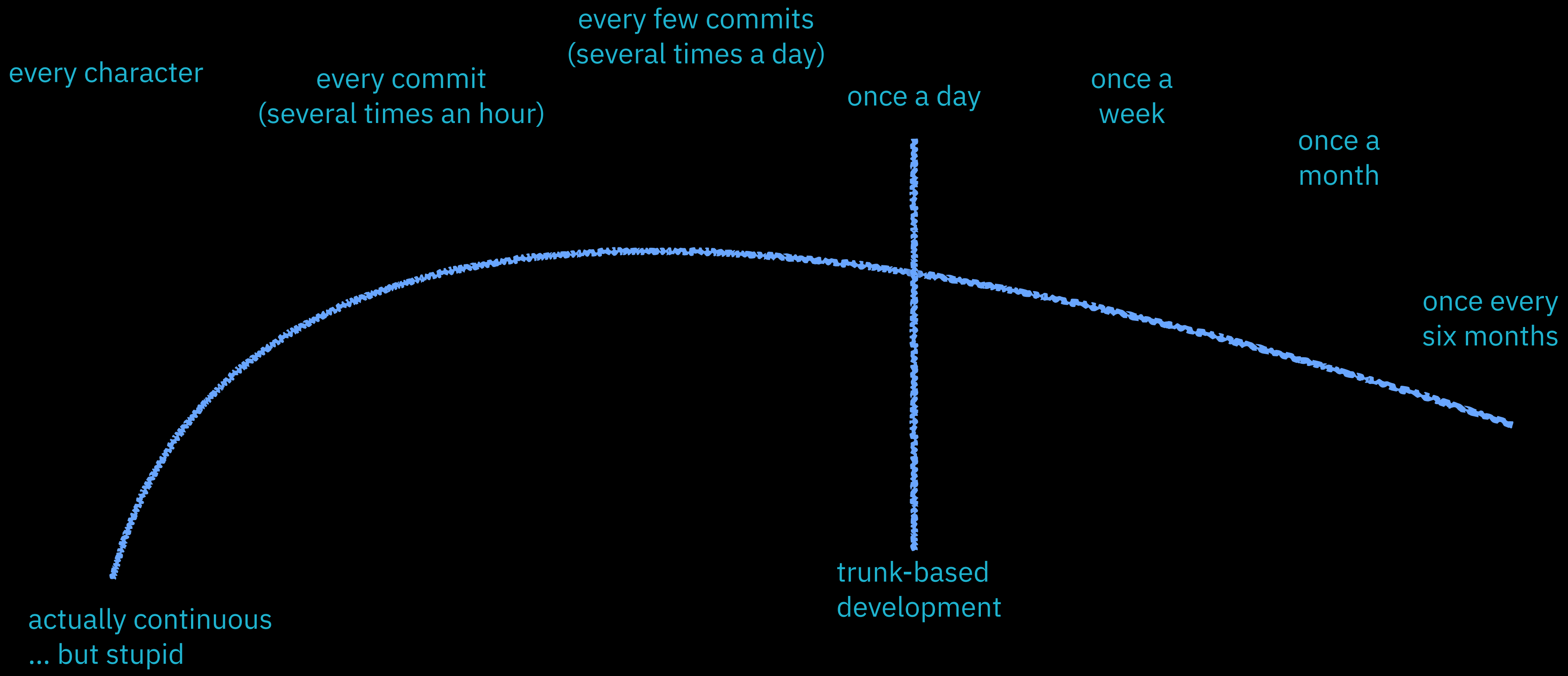
how often should you integrate?



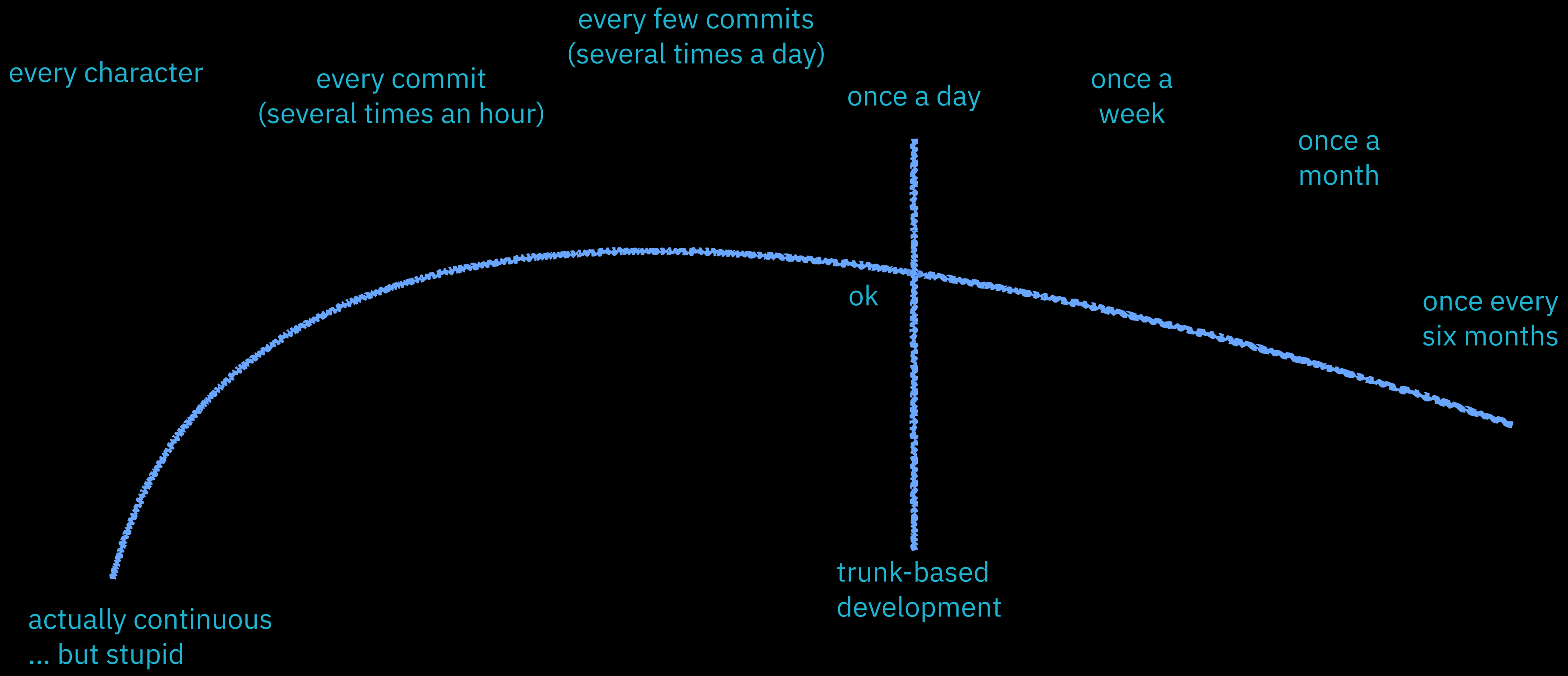
how often should you integrate?



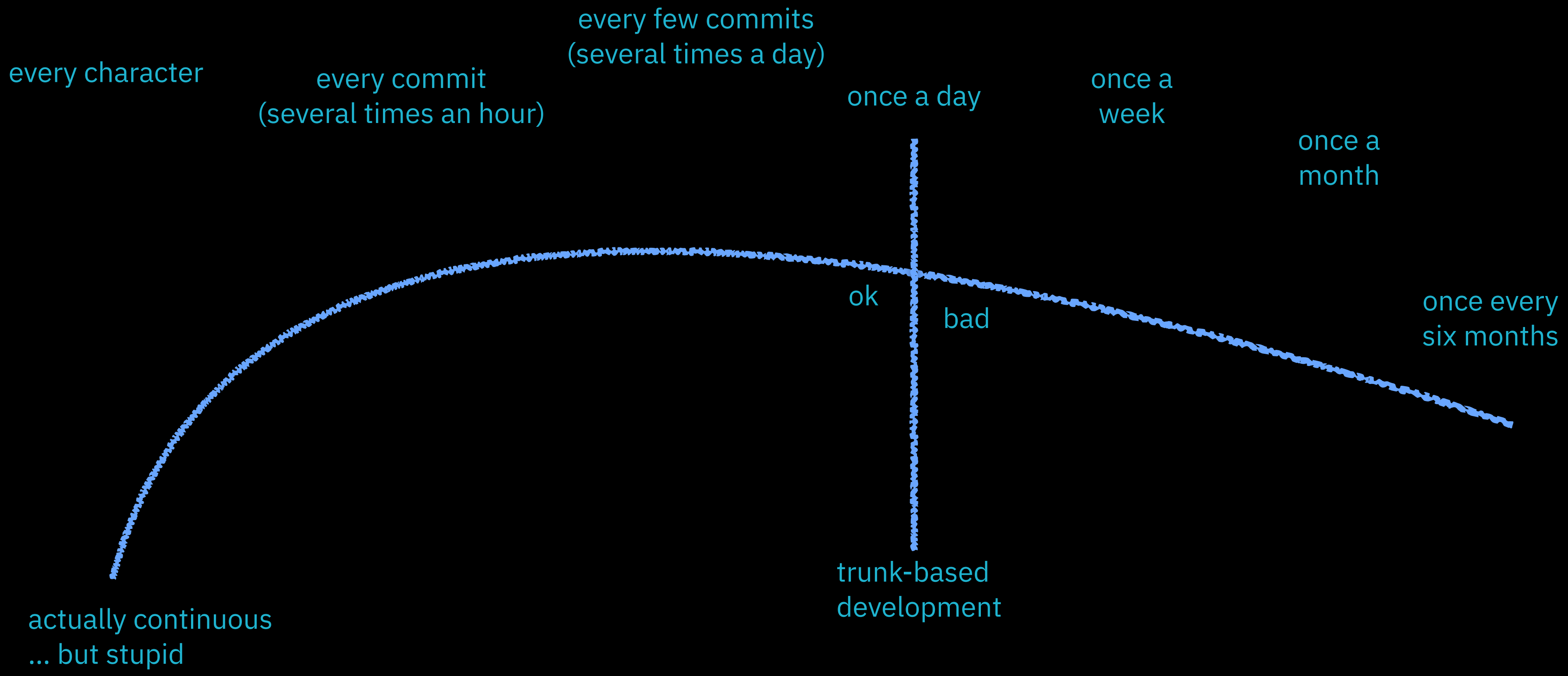
how often should you integrate?


every character
every commit (several times an hour)
every few commits (several times a day)
once a day
once a week
once a month
once every six months
actually continuous ... but stupid
trunk-based development

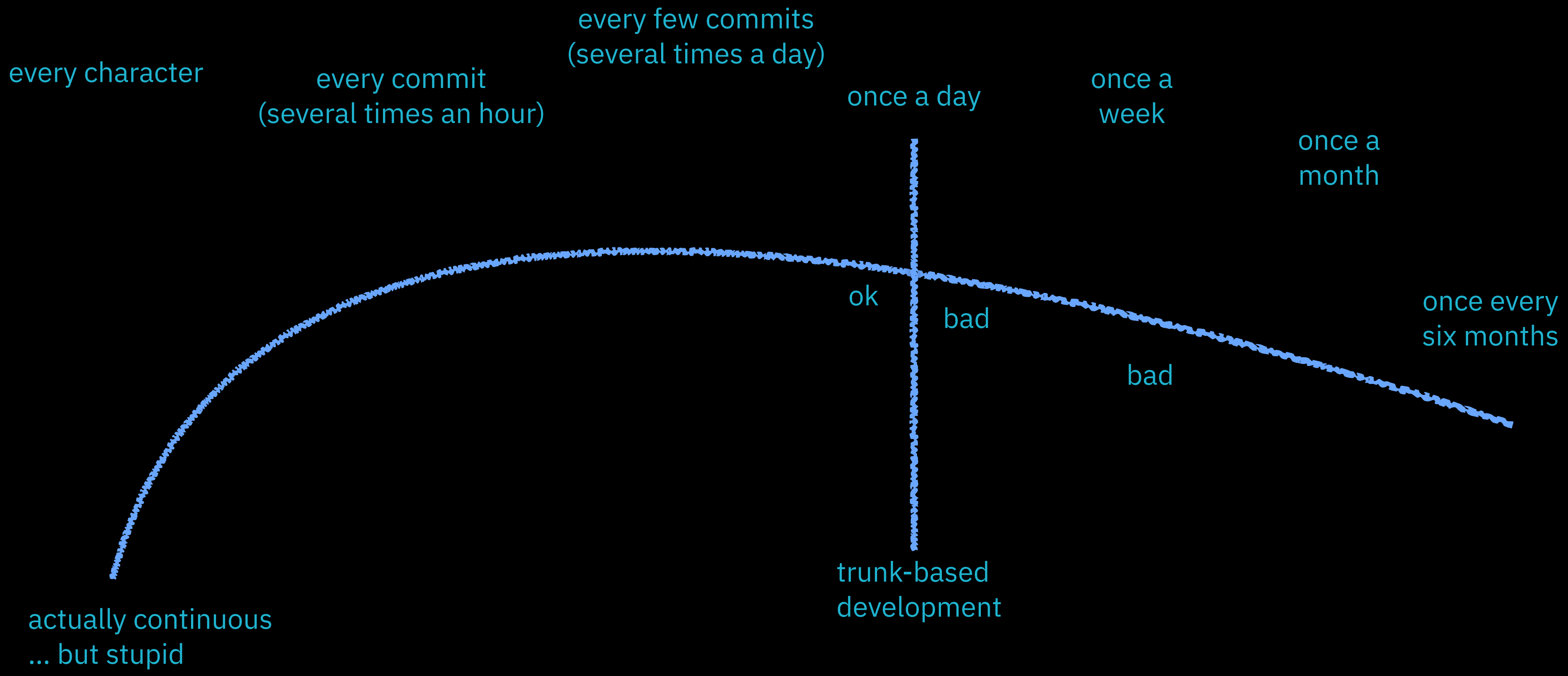
how often should you integrate?



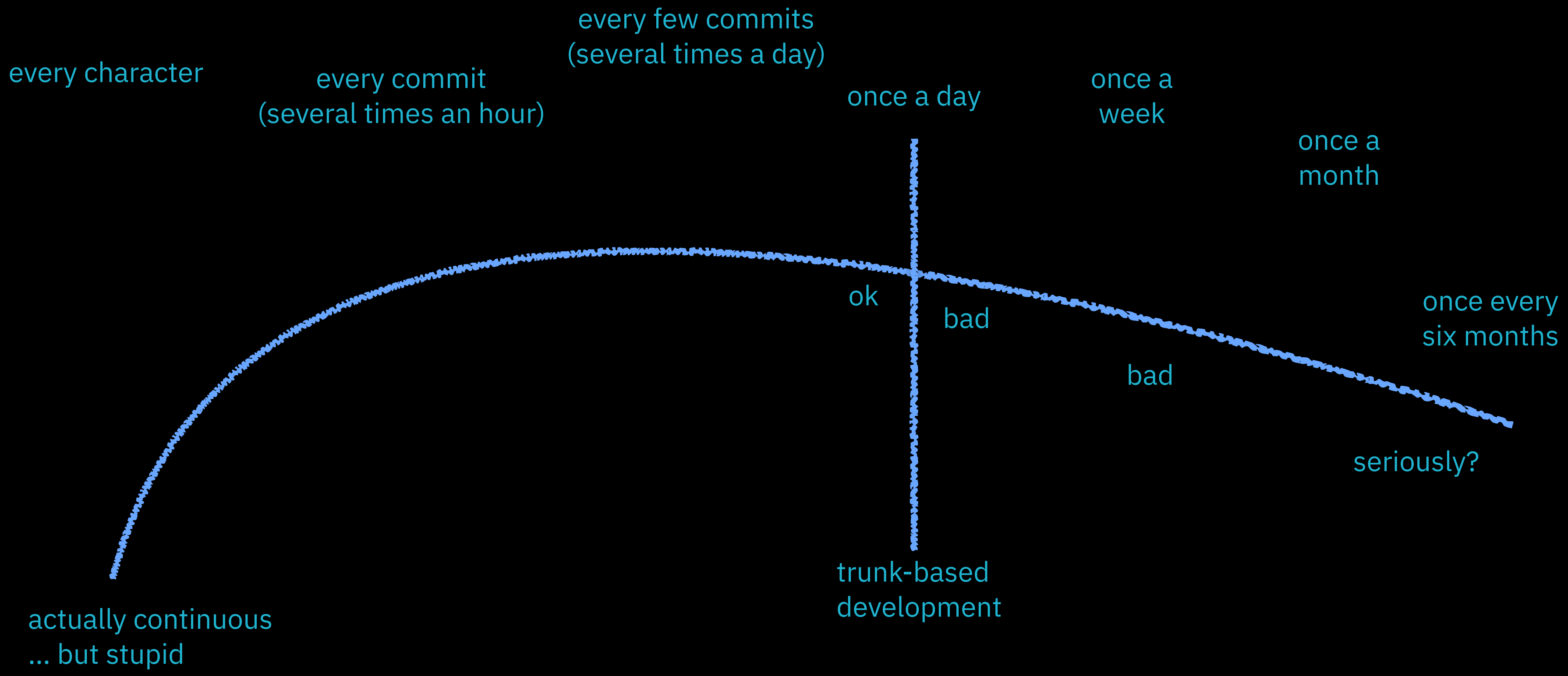
how often should you integrate?



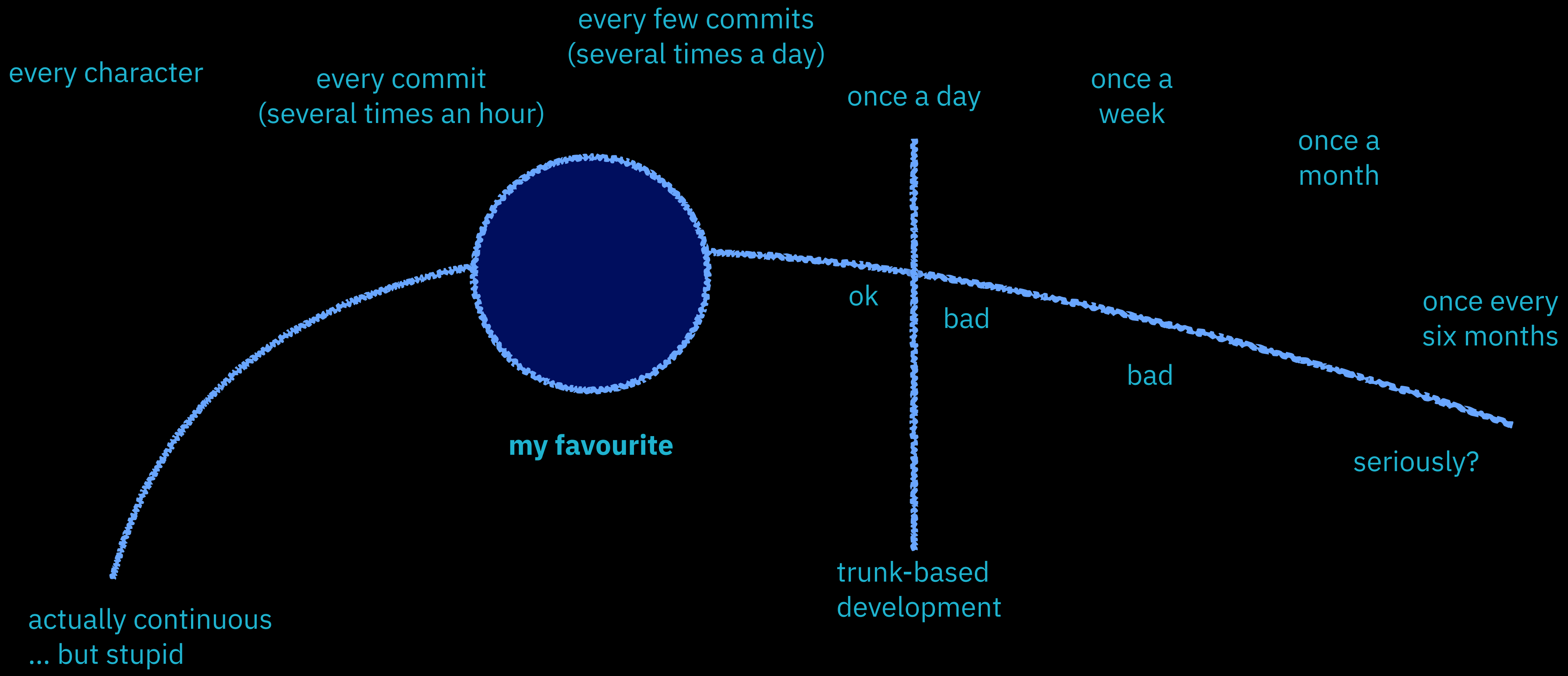
how often should you integrate?


every character
every commit (several times an hour)
every few commits (several times a day)
once a day
once a week
once a month
ok
bad
bad
once every six months
trunk-based development
actually continuous ... but stupid

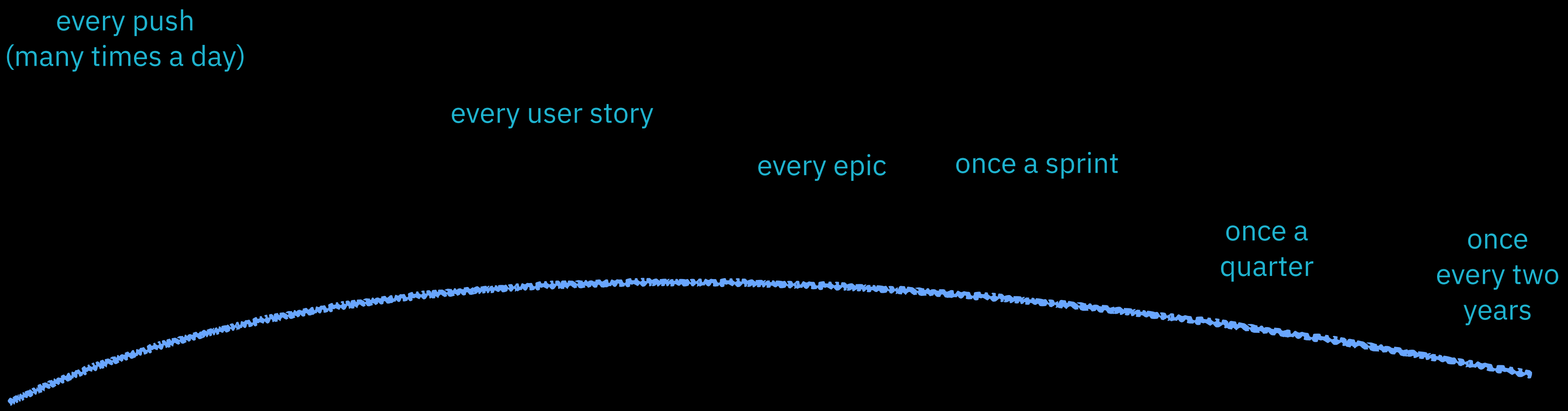
how often should you integrate?



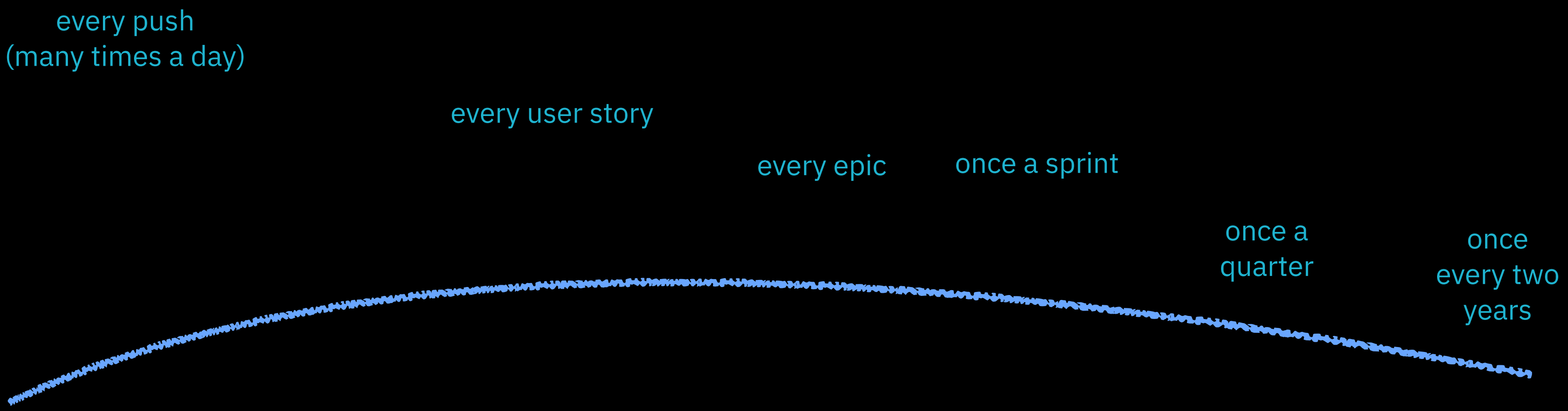
how often should you integrate?



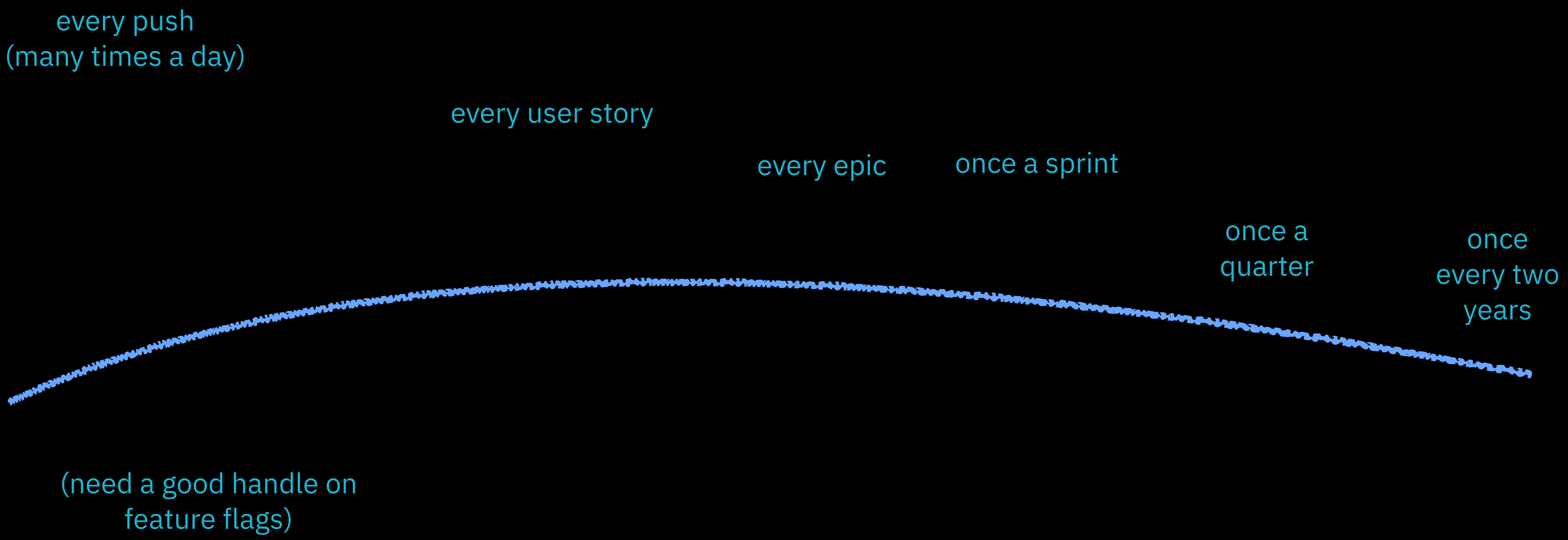
how often should you release?



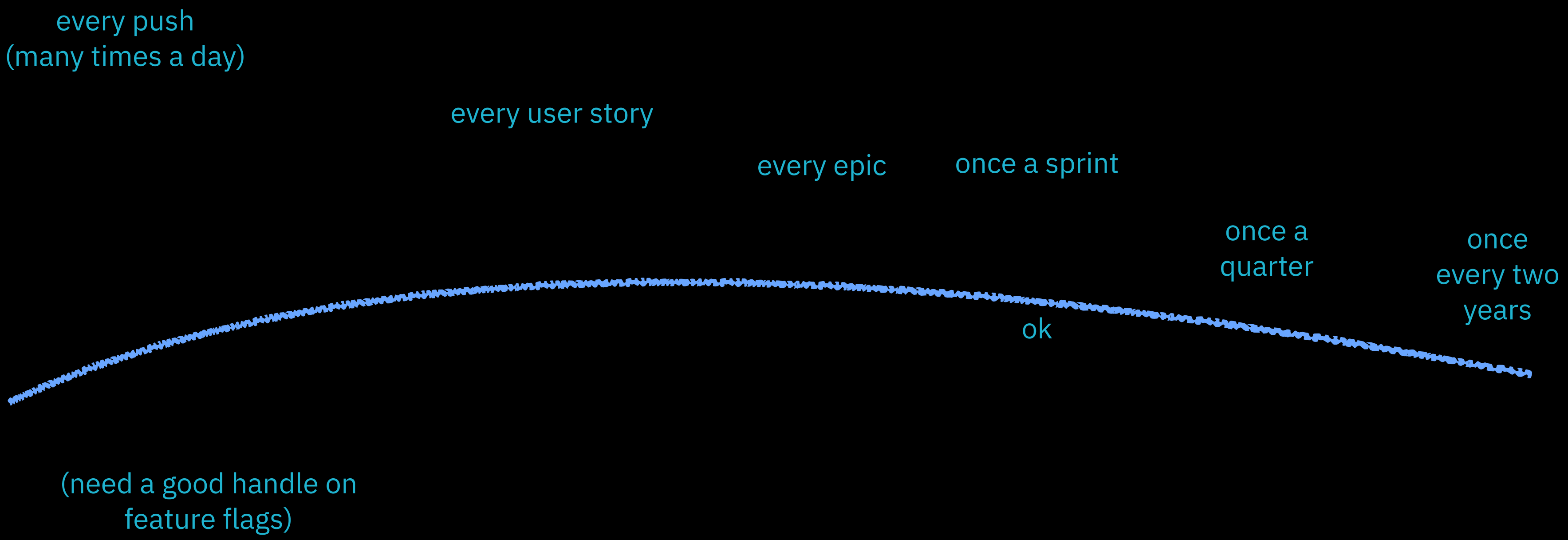
how often should you deploy?



how often should you deploy?



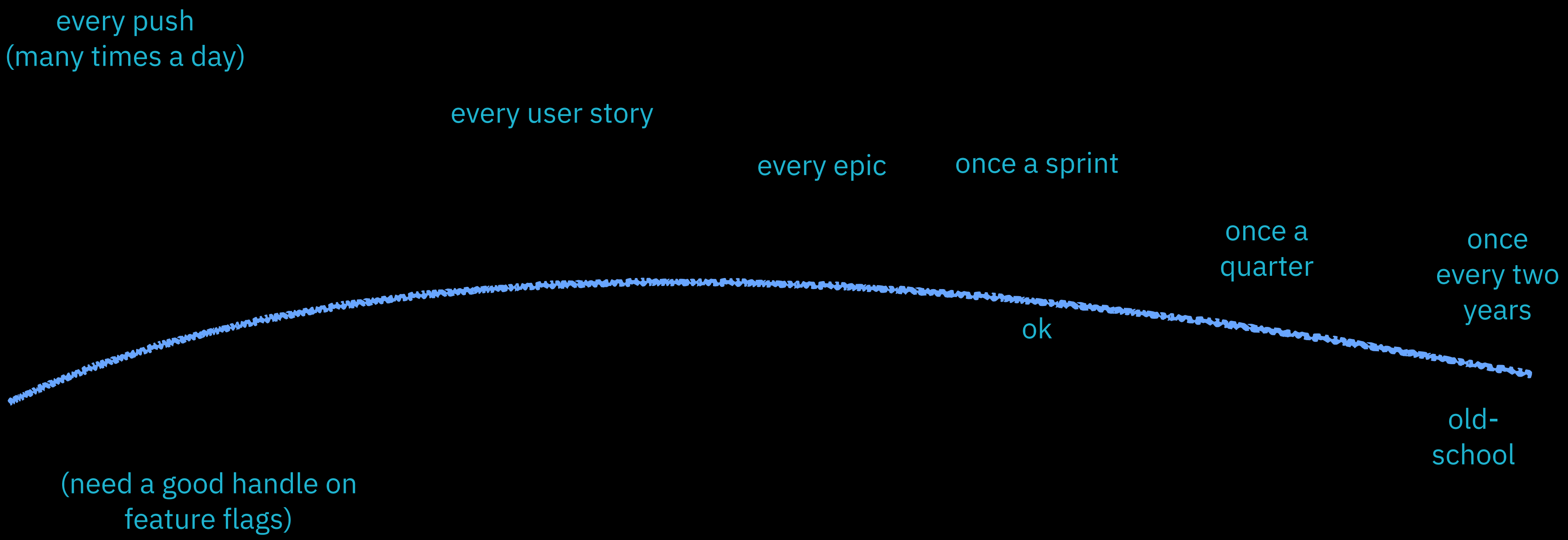
how often should you deploy?



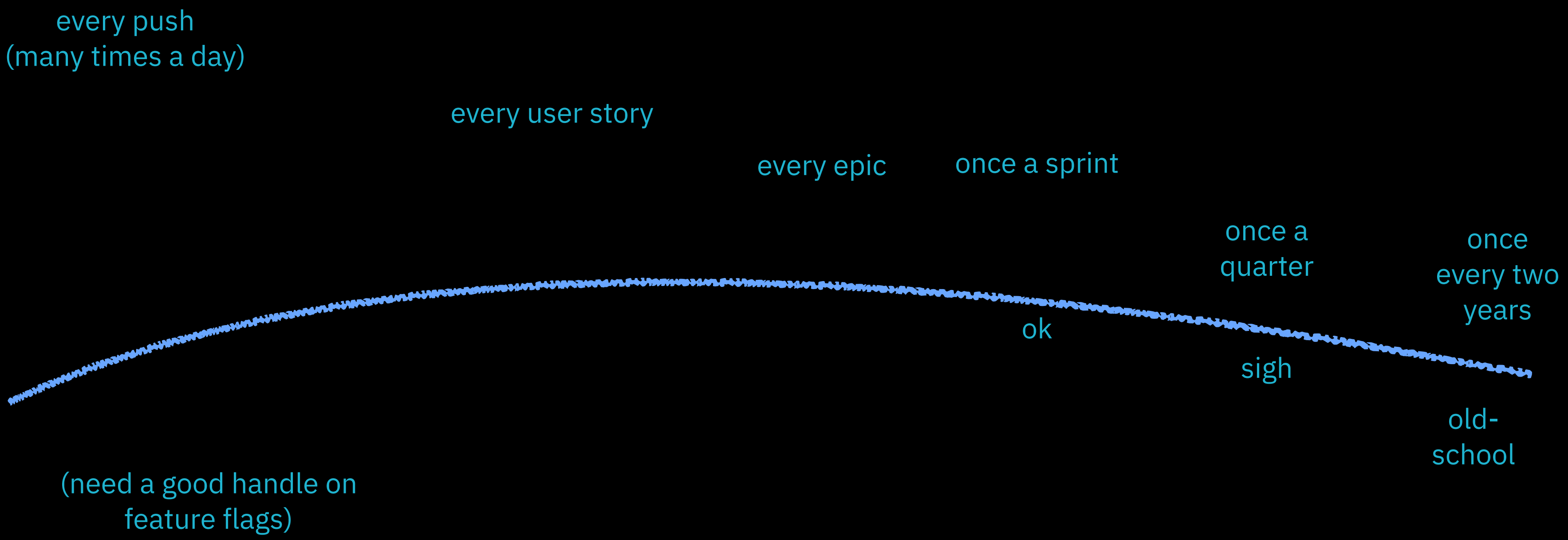
#IBMGarage

@holly_cummins

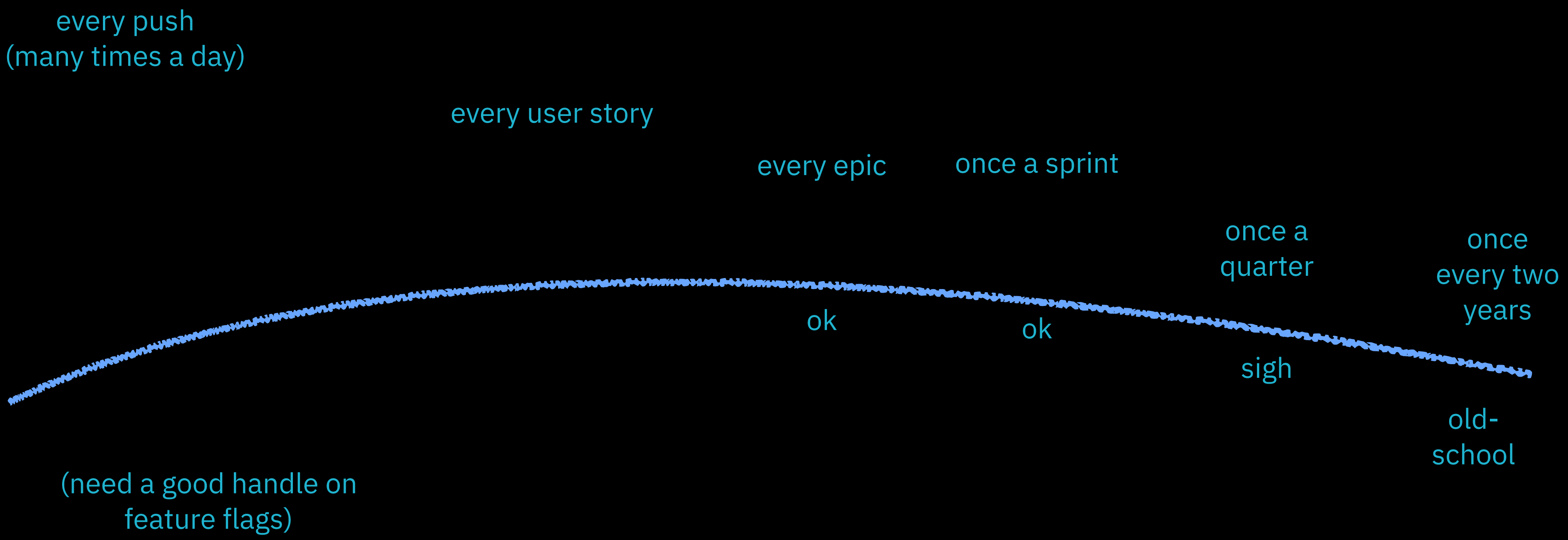
how often should you deploy?



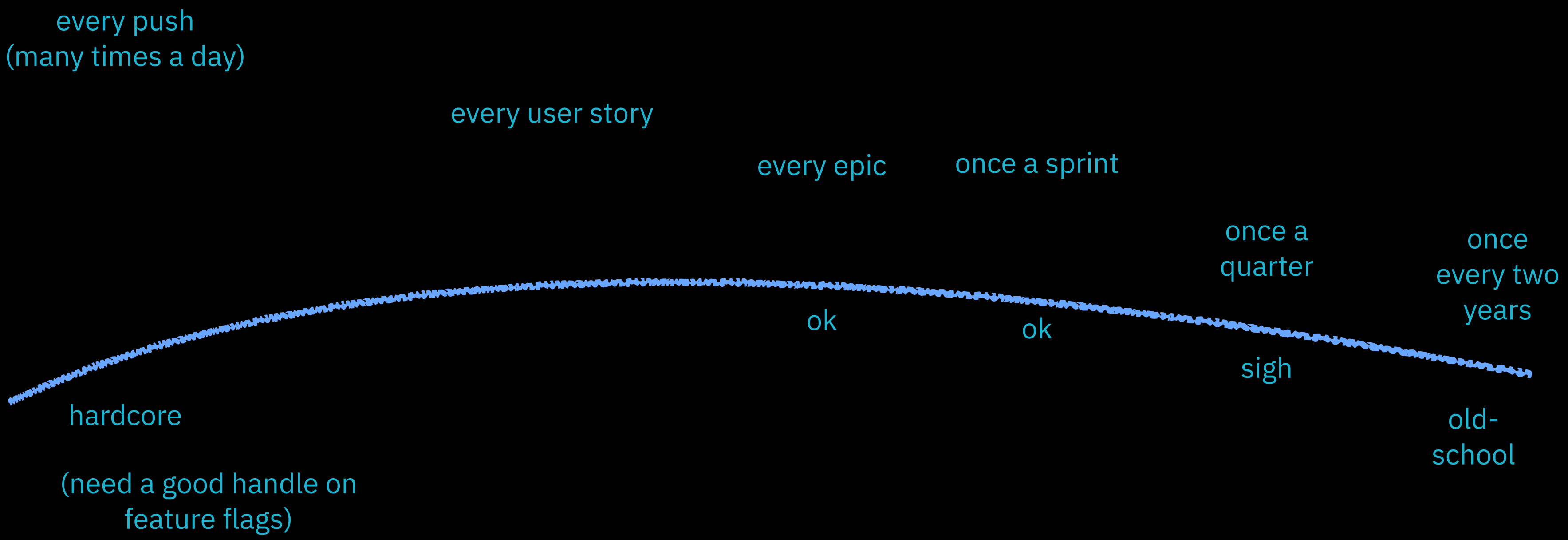
how often should you deploy?



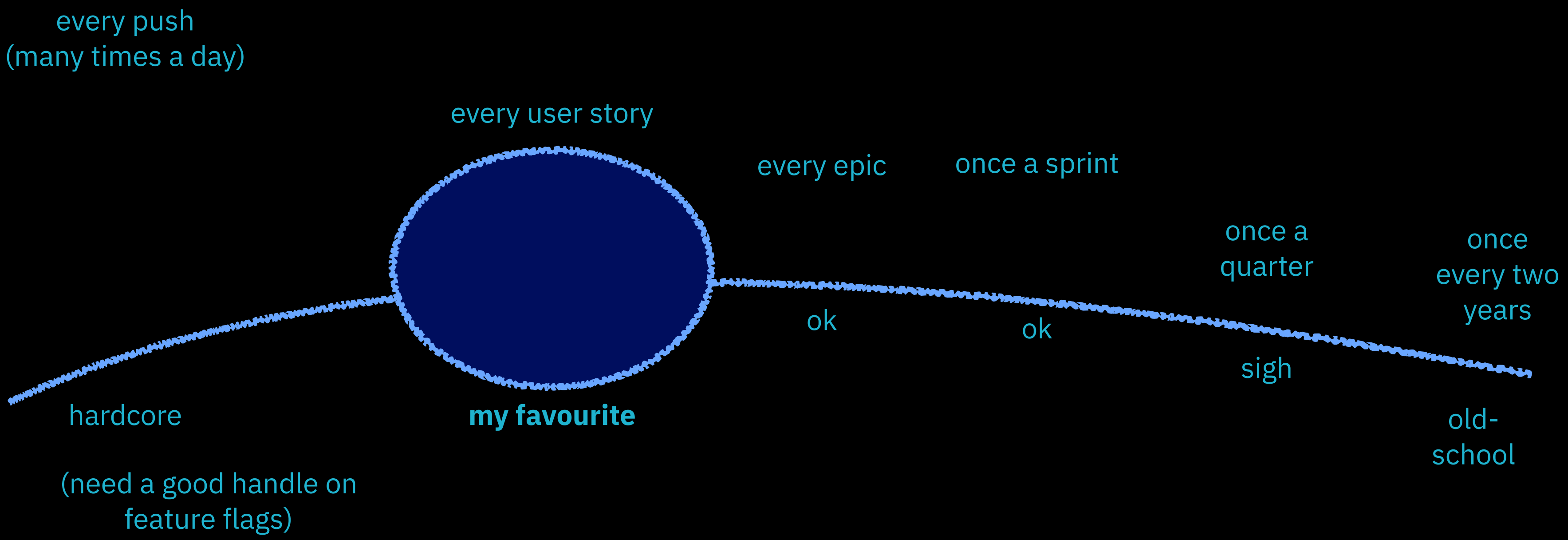
how often should you deploy?



how often should you deploy?



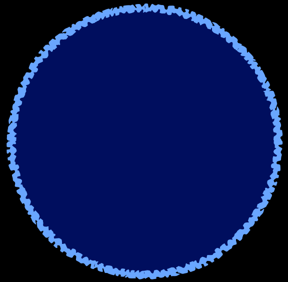
how often should you deploy?



how often should you test in staging?

how often should you deliver?

how often should you deliver?



“we can’t actually **release** this.”

why?

what’s stopping more frequent deploys?

“we can’t release this microservice…

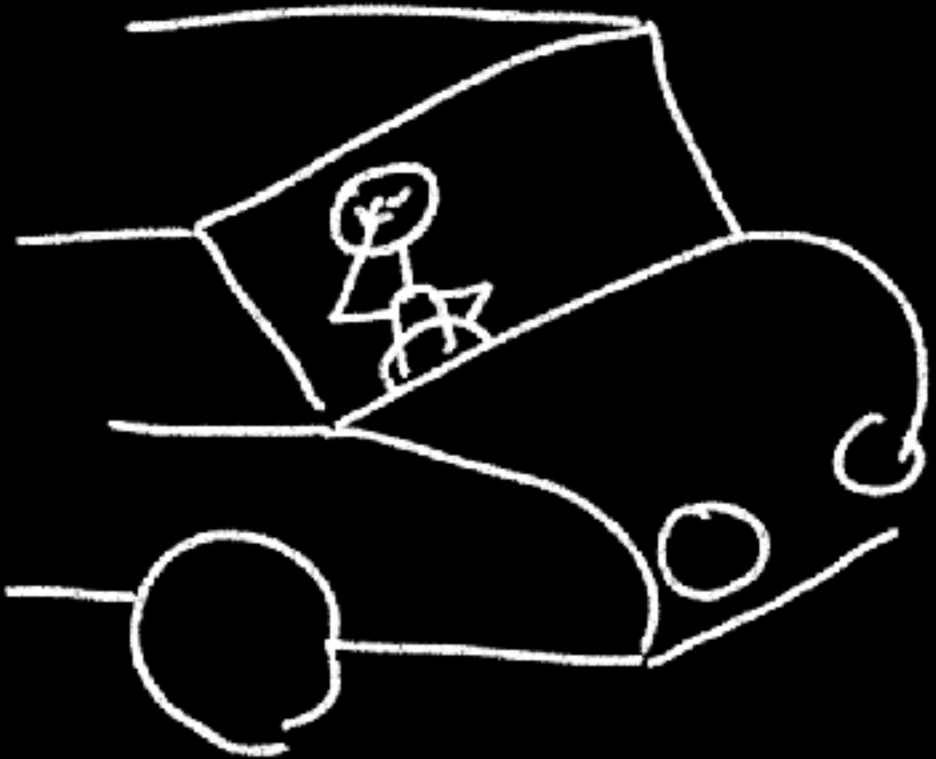
we deploy all our microservices at the same time.”

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

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

- Reid Hoffman

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

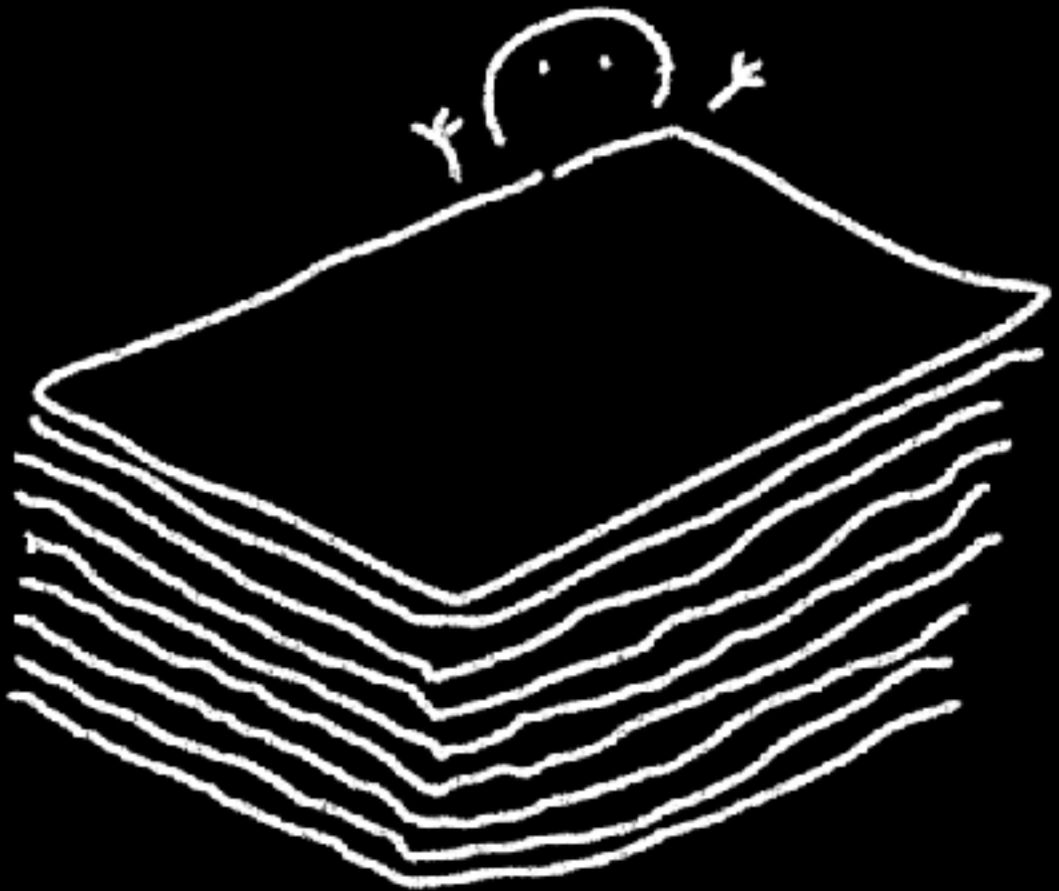
feature flags

A/B testing

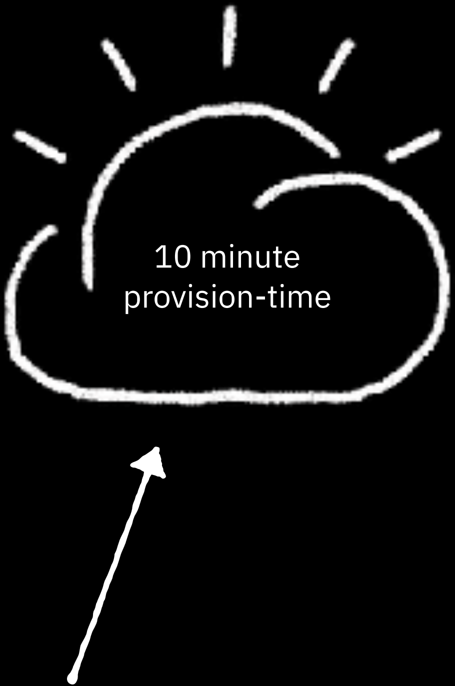
canary deploys

fail 4

the locked-
down totally
rigid inflexible
un-cloudy cloud

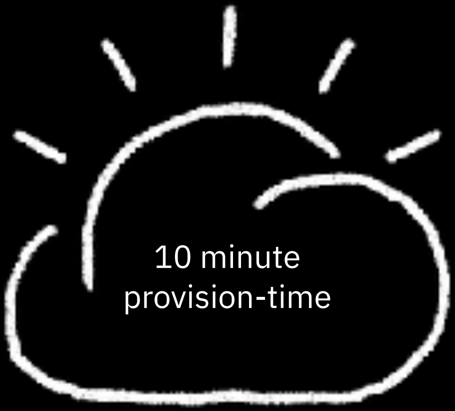


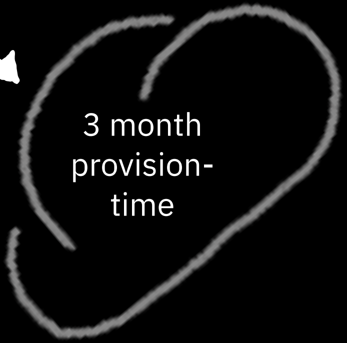
“we’ve scheduled the architecture board review for a month after the project is ready to ship” “this provisioning software is broken”



what we sold

"this provisioning software is broken"

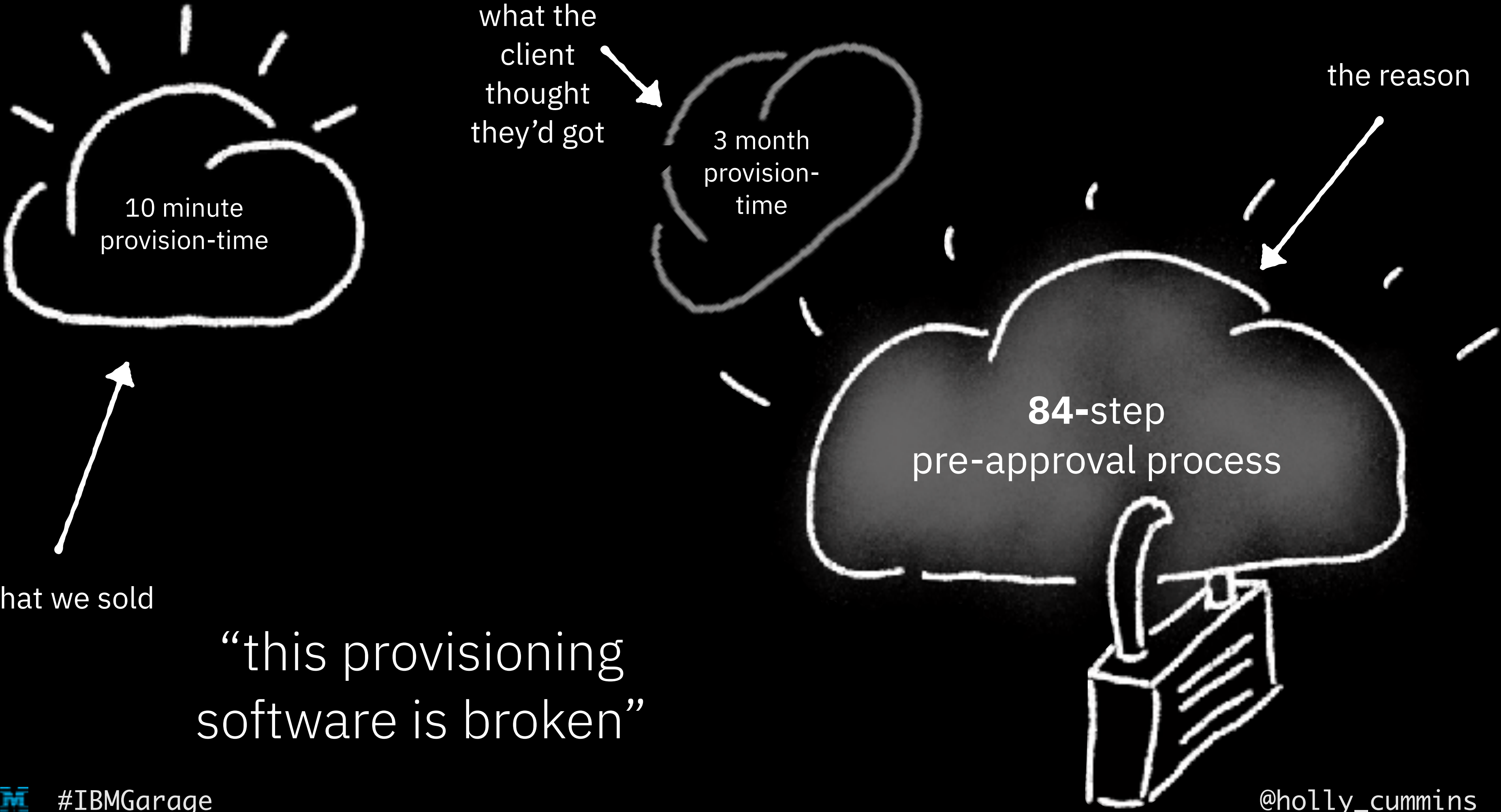




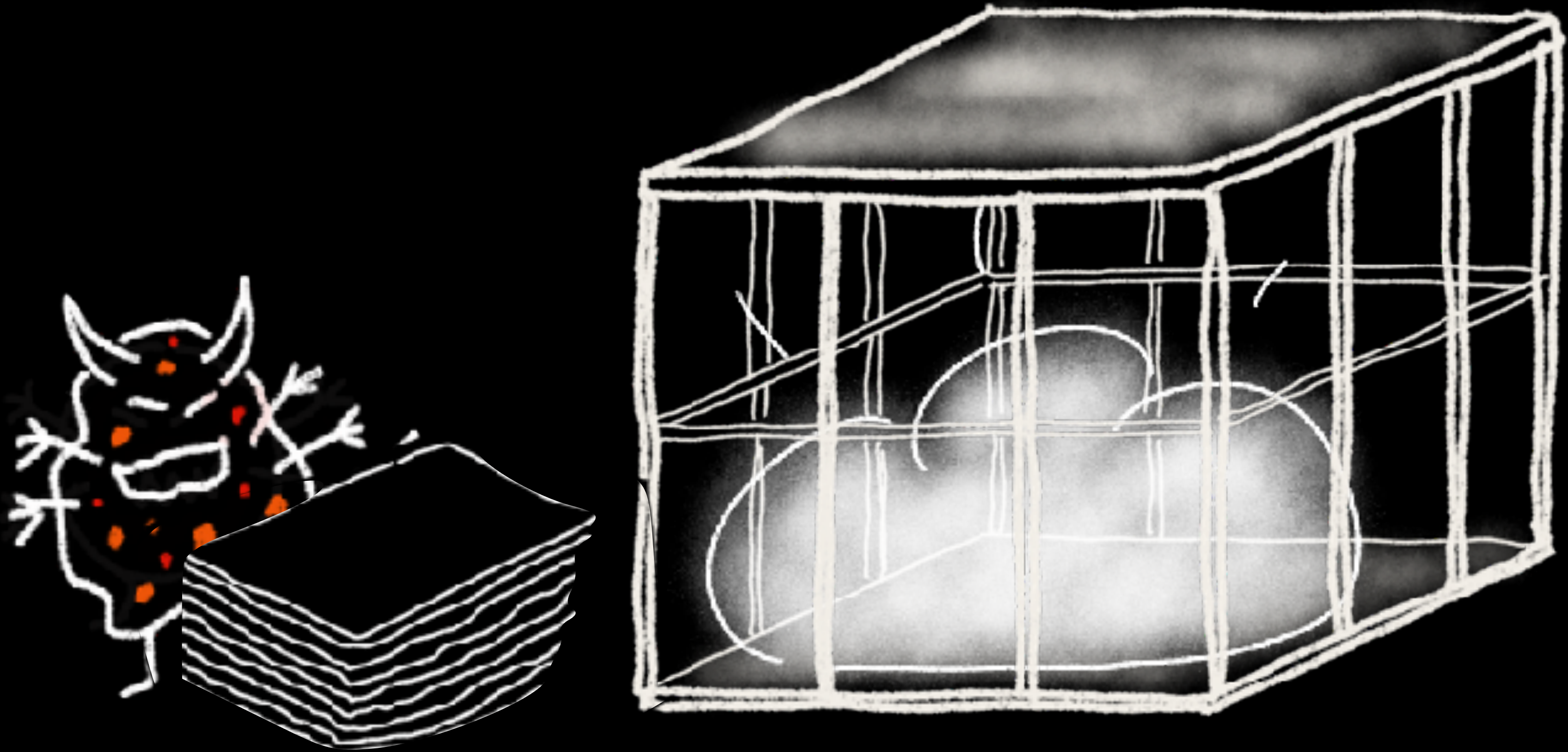
what the
client
thought
they'd got

what we sold

“this provisioning
software is broken”



"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

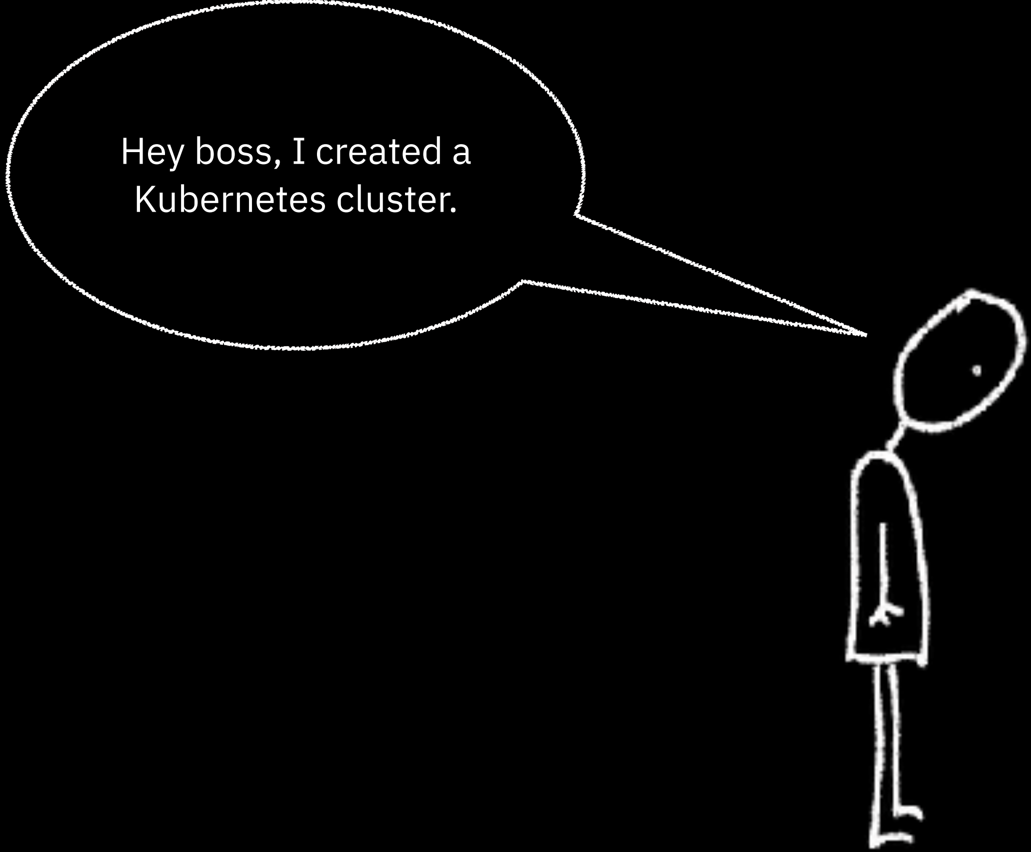
fail 5

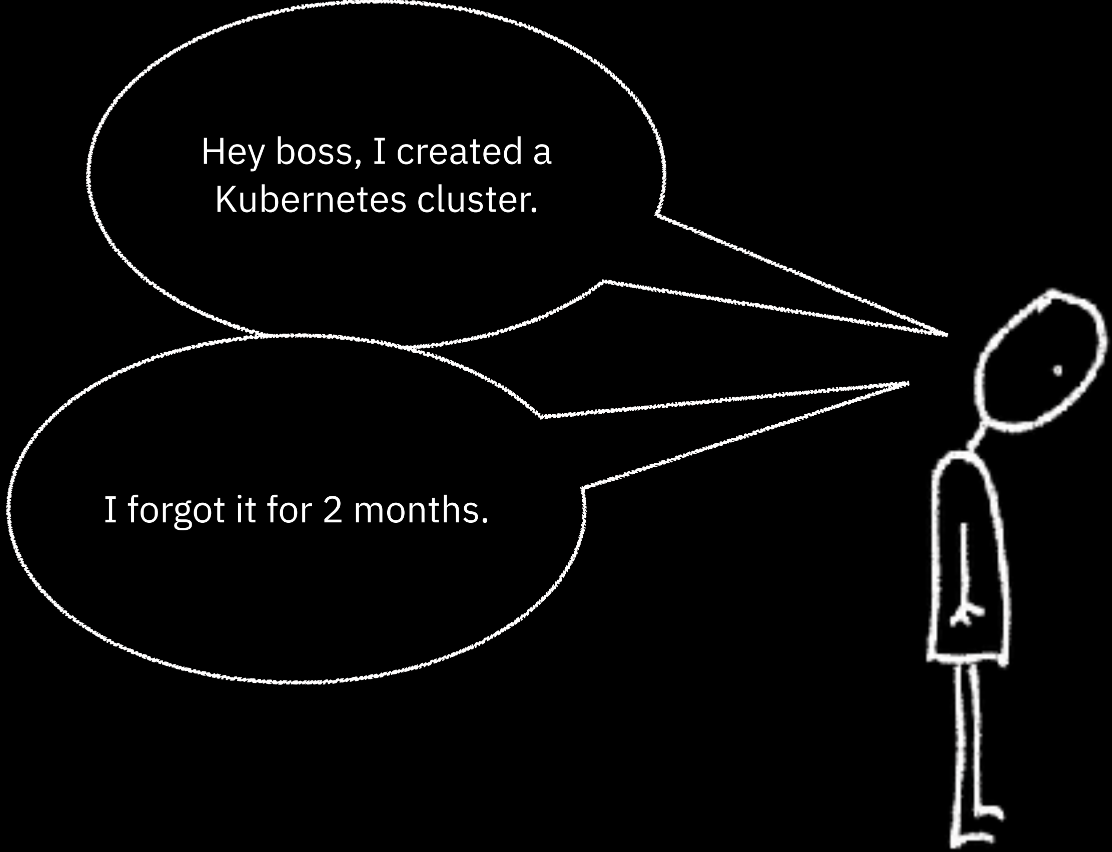
the mystery money pit

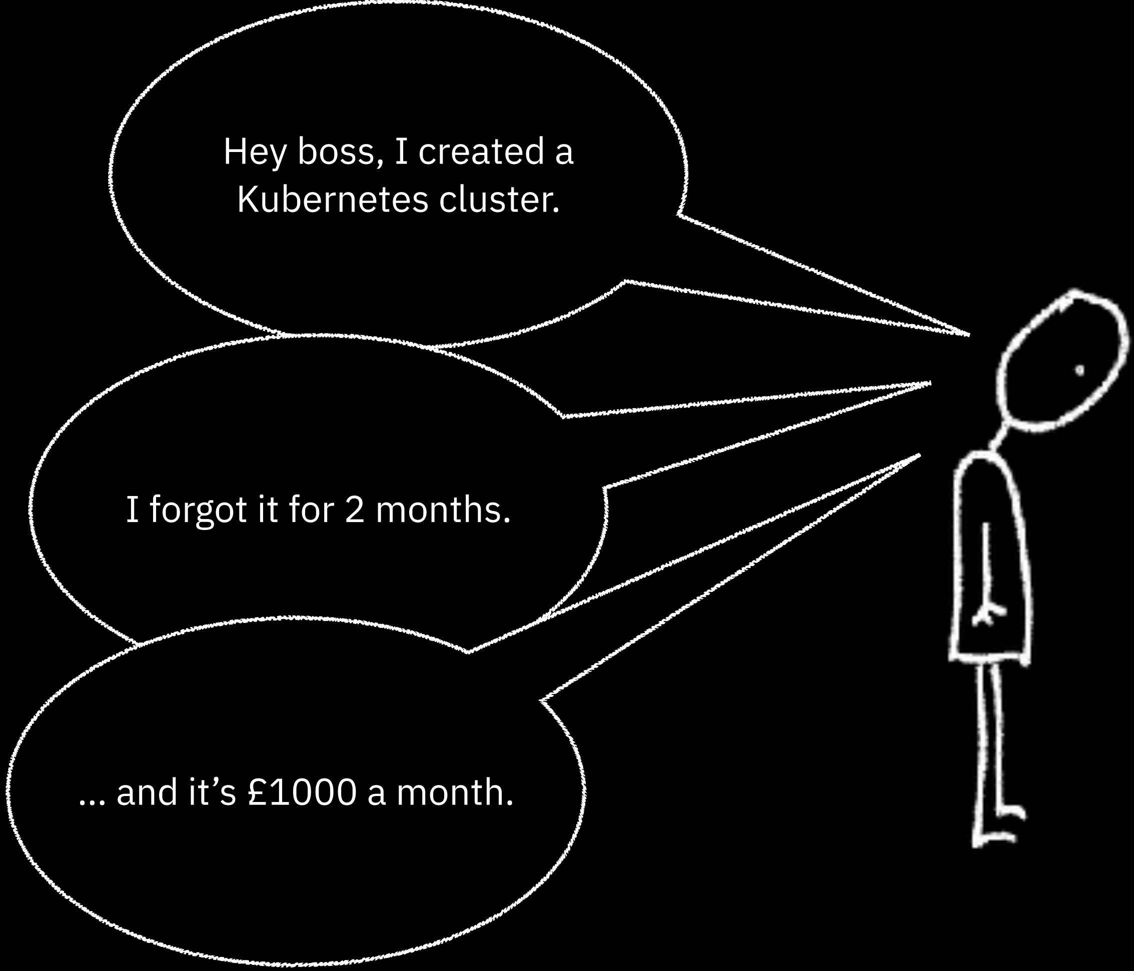
the cloud makes it so **easy** to provision hardware.

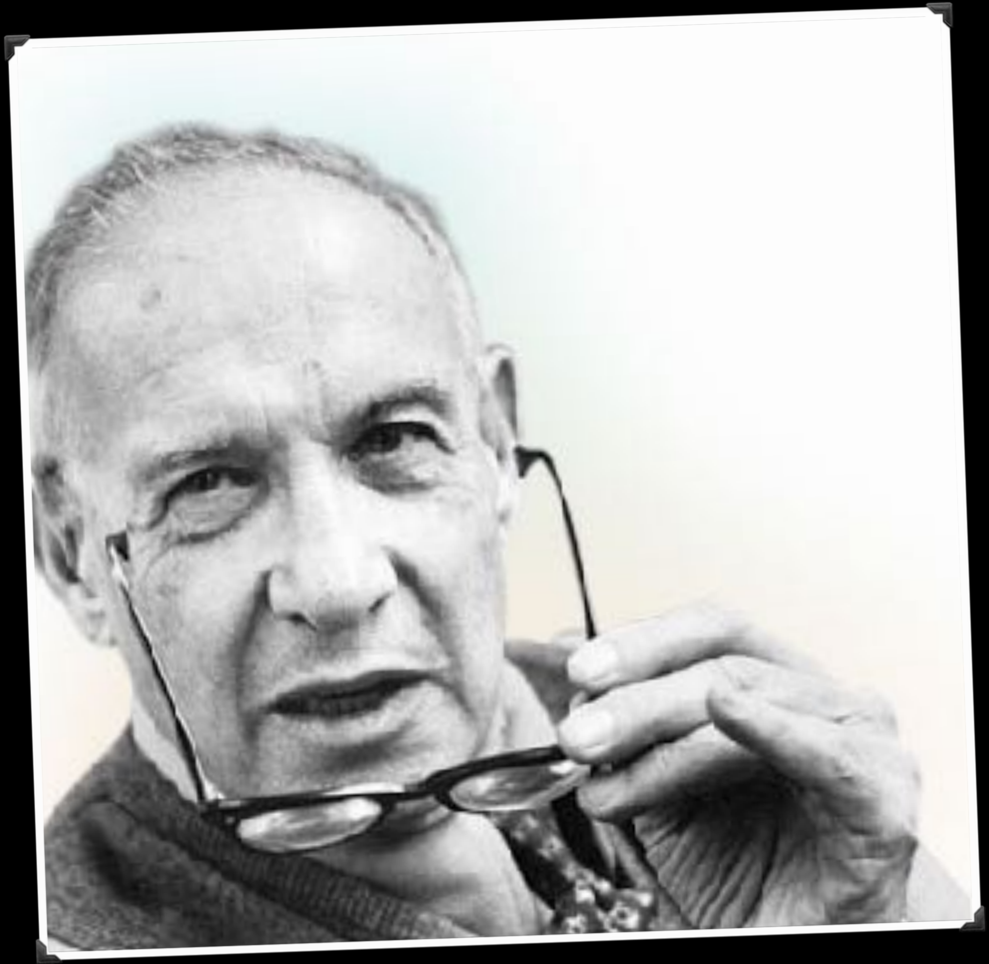
that doesn’t mean the hardware is free.

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.



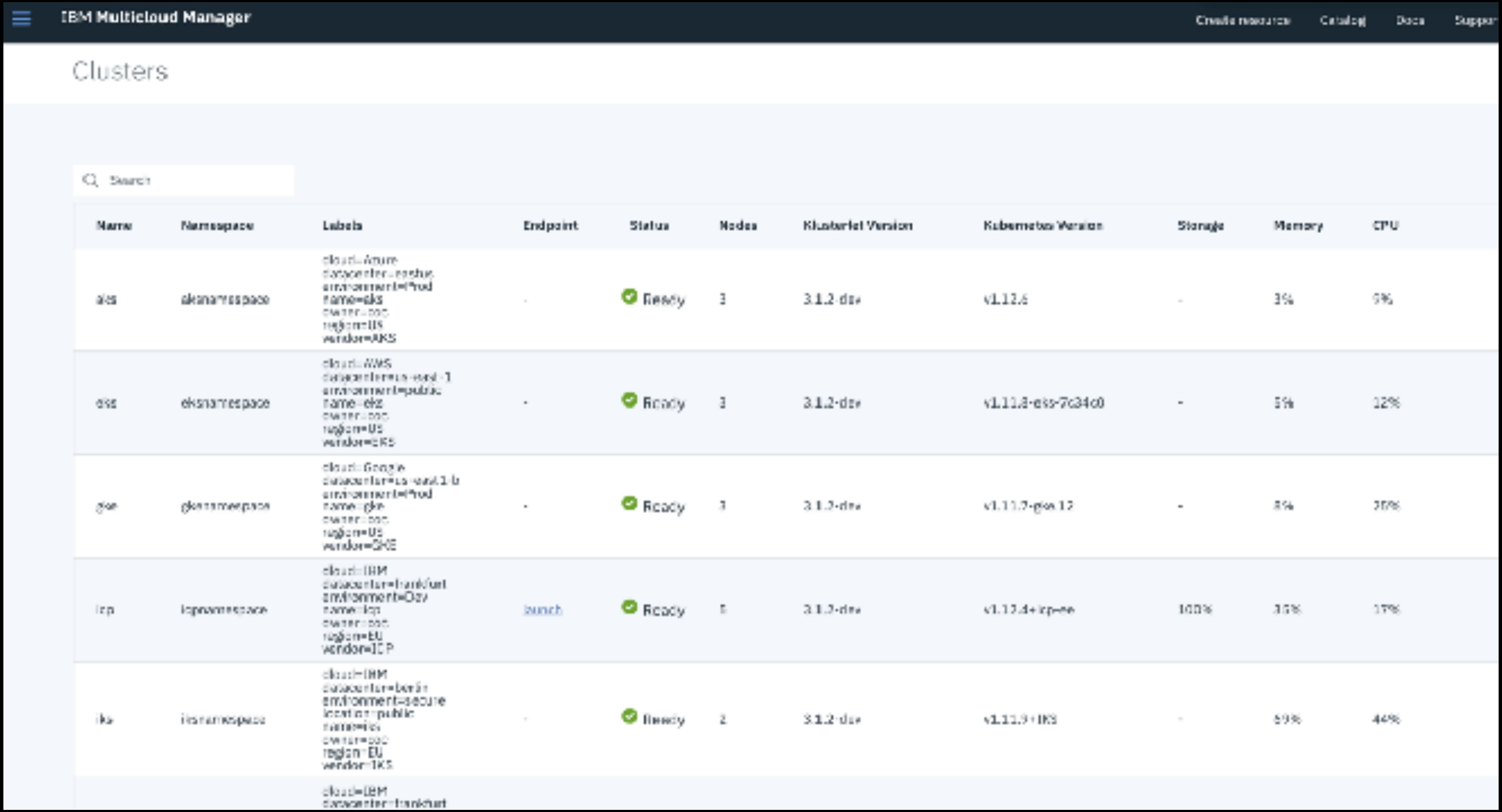
There is surely nothing quite so useless as doing with great efficiency what should not be done at all.

— Peter Drucker

ouch

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

cloud to manage your clouds


IBM Multicloud Manager   Create resource   Catalog   Docs   Suppor

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=[illegible] 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=[illegible] 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=[illegible] region=US vendor=GKE	-	✓ Ready	3	3.1.2-dev	v1.11.7-gke.12	-	8%	20%
icp	icpnamespace	cloud=IBM datacenter=frankfurt environment=Dev name=icp owner=[illegible] 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=[illegible] region=EU vendor=IKS	-	✓ Ready	2	3.1.2-dev	v1.11.9+IKS	-	59%	44%
		cloud=IBM datacenter=frankfurt								

fail 6

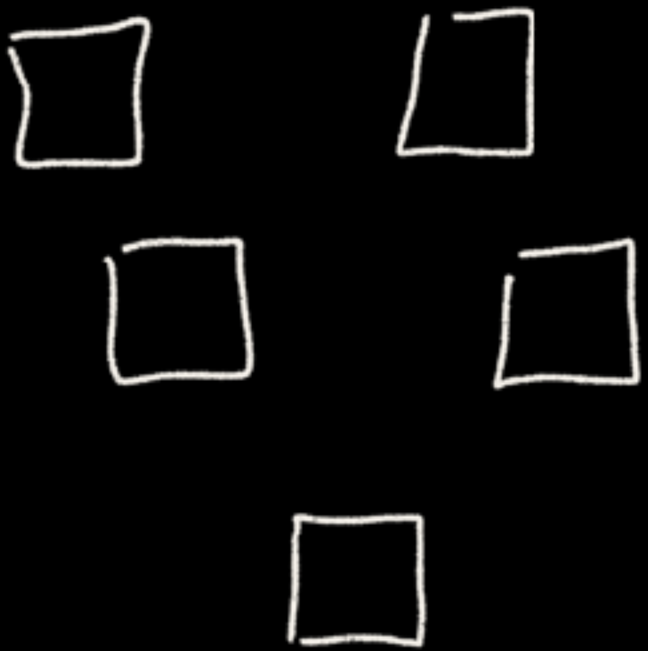
cloud-native spaghetti

“every time we change one microservice, another breaks”

distributed != decoupled

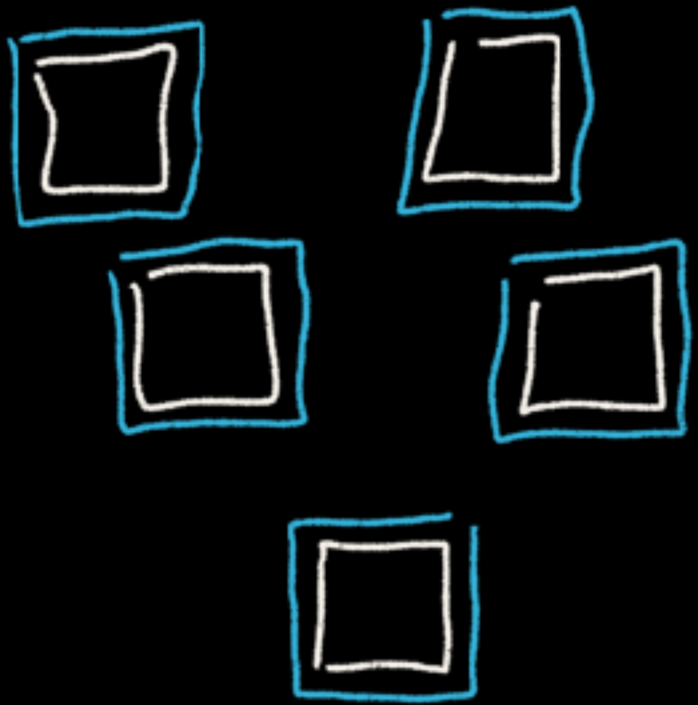
“each of our microservices has duplicated the same object model … with twenty classes and seventy fields”

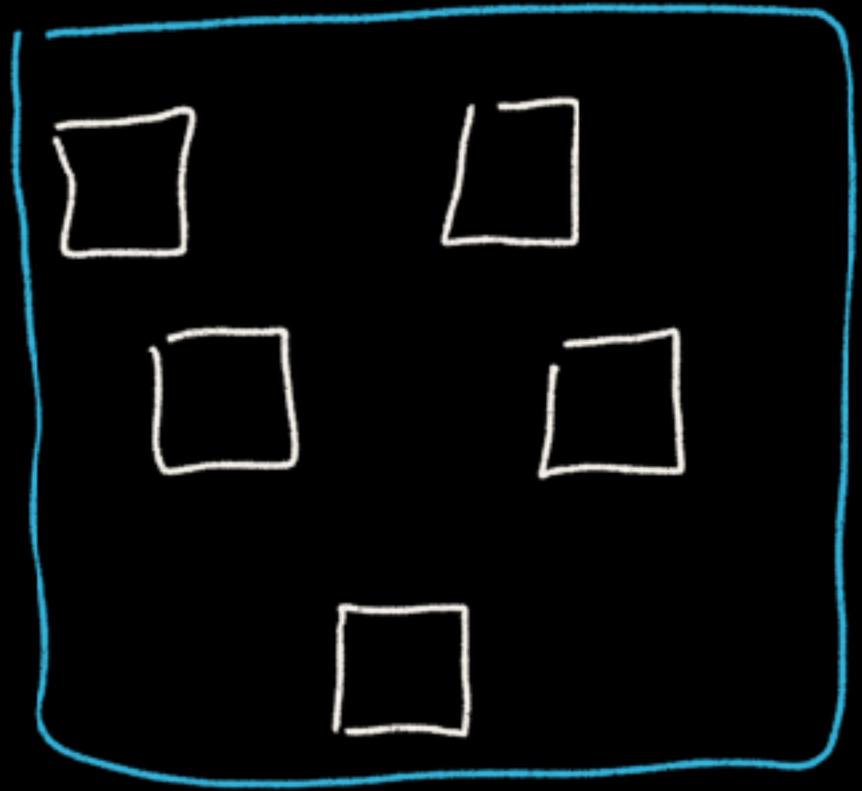
Microservice



Microservice

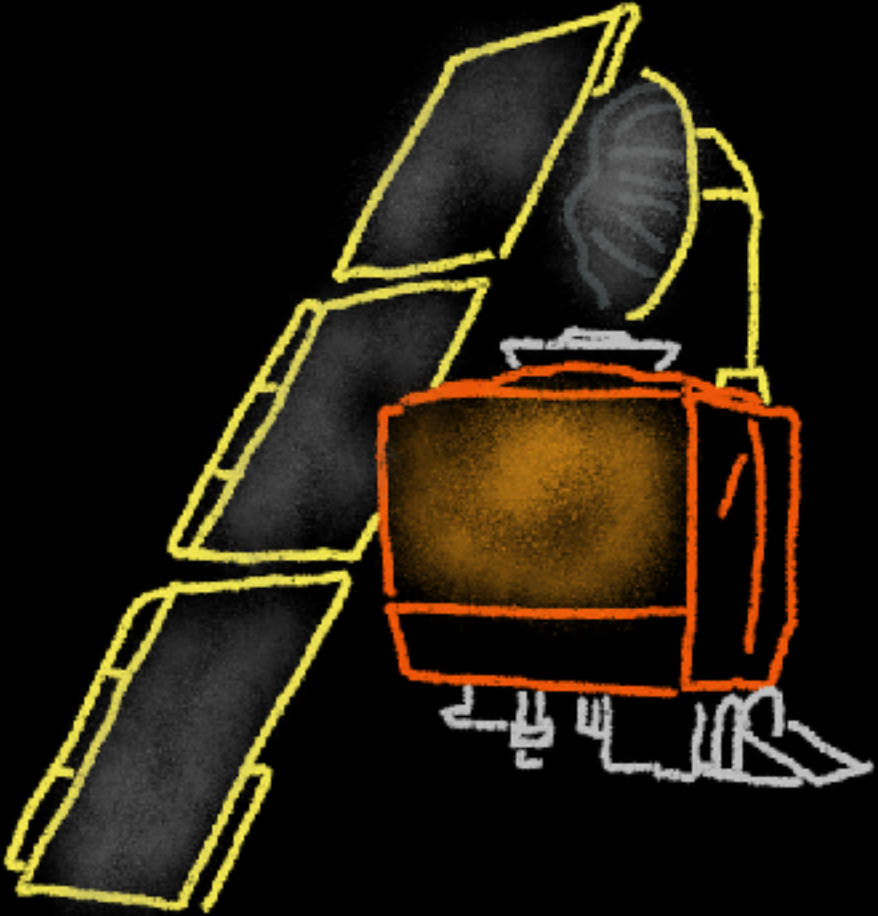
Domain

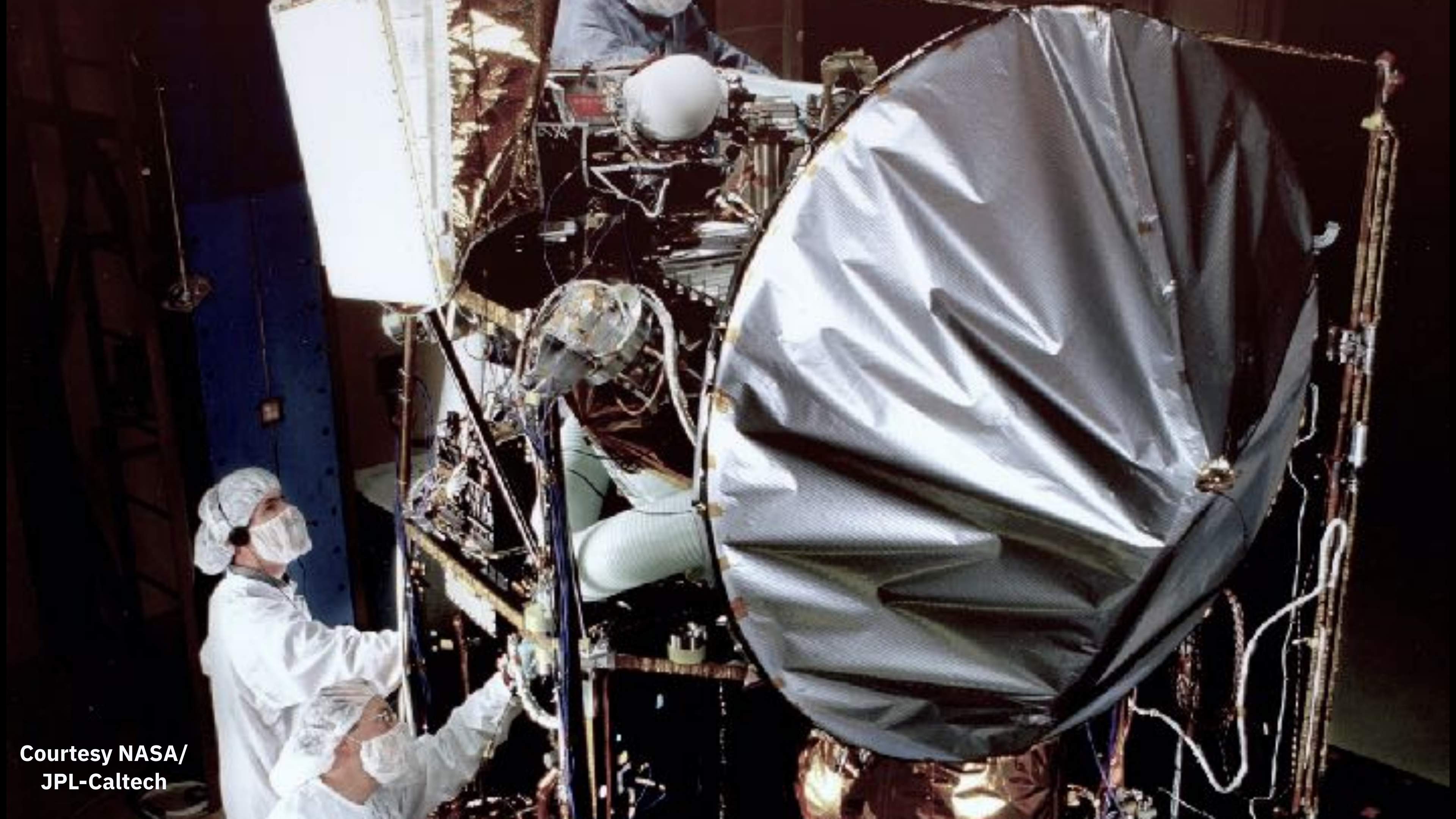


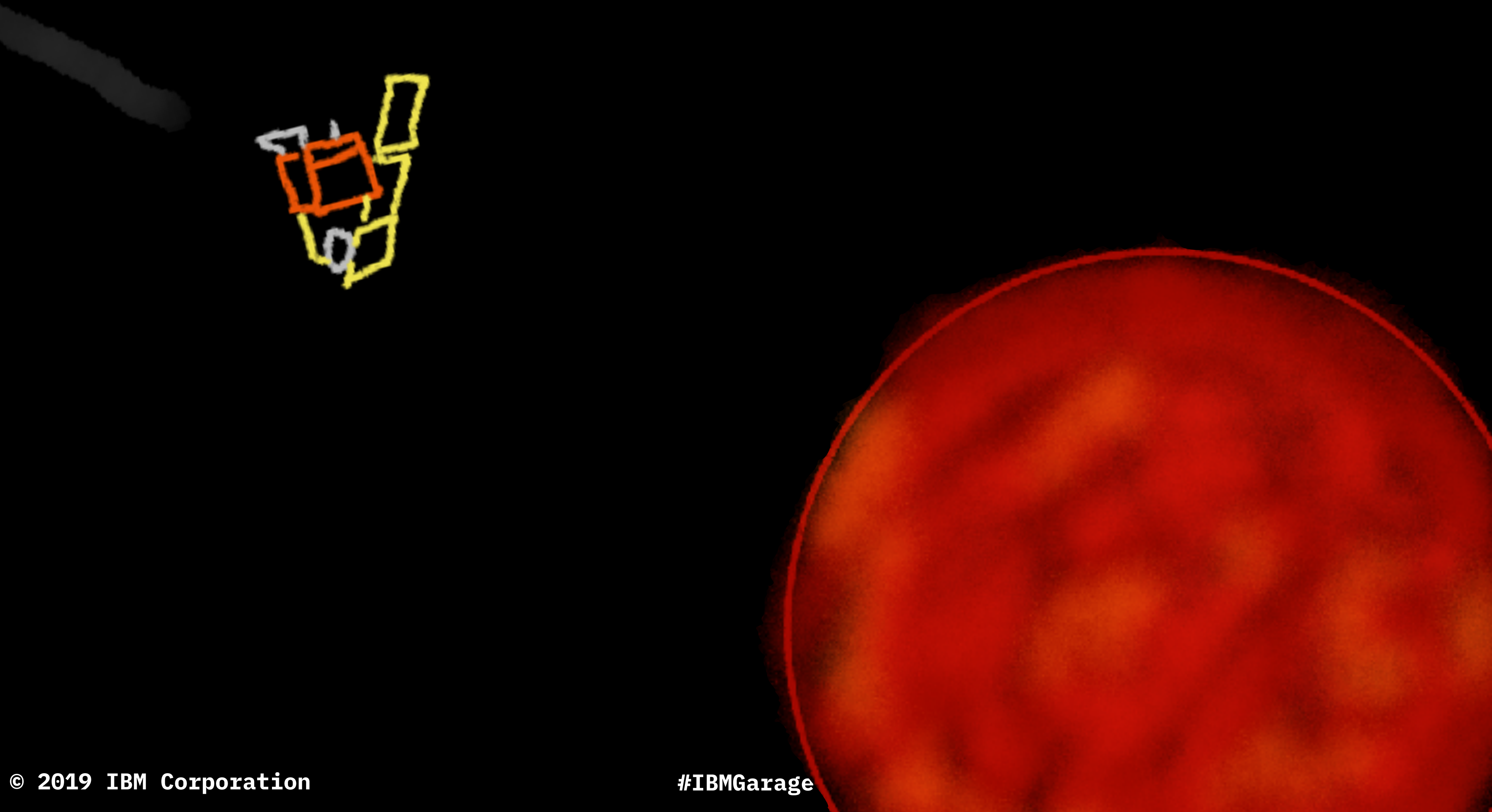


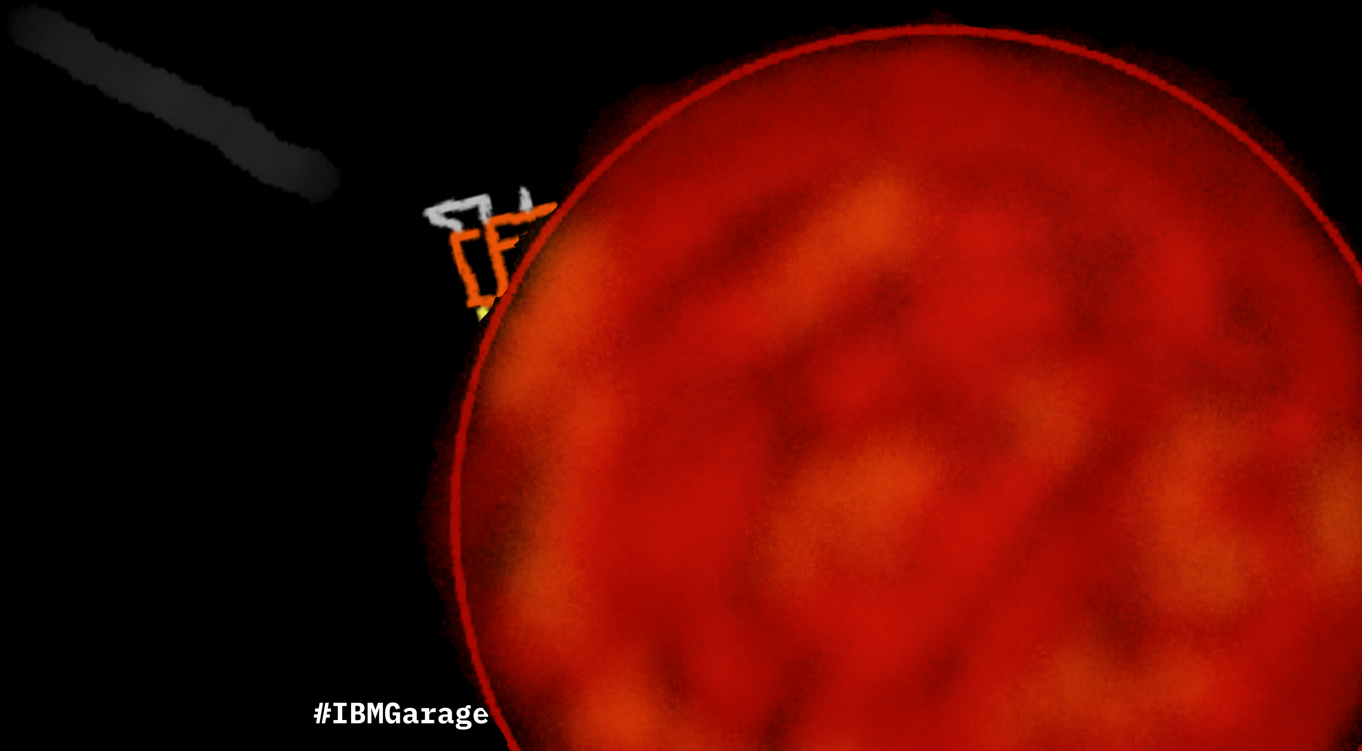
Microservice

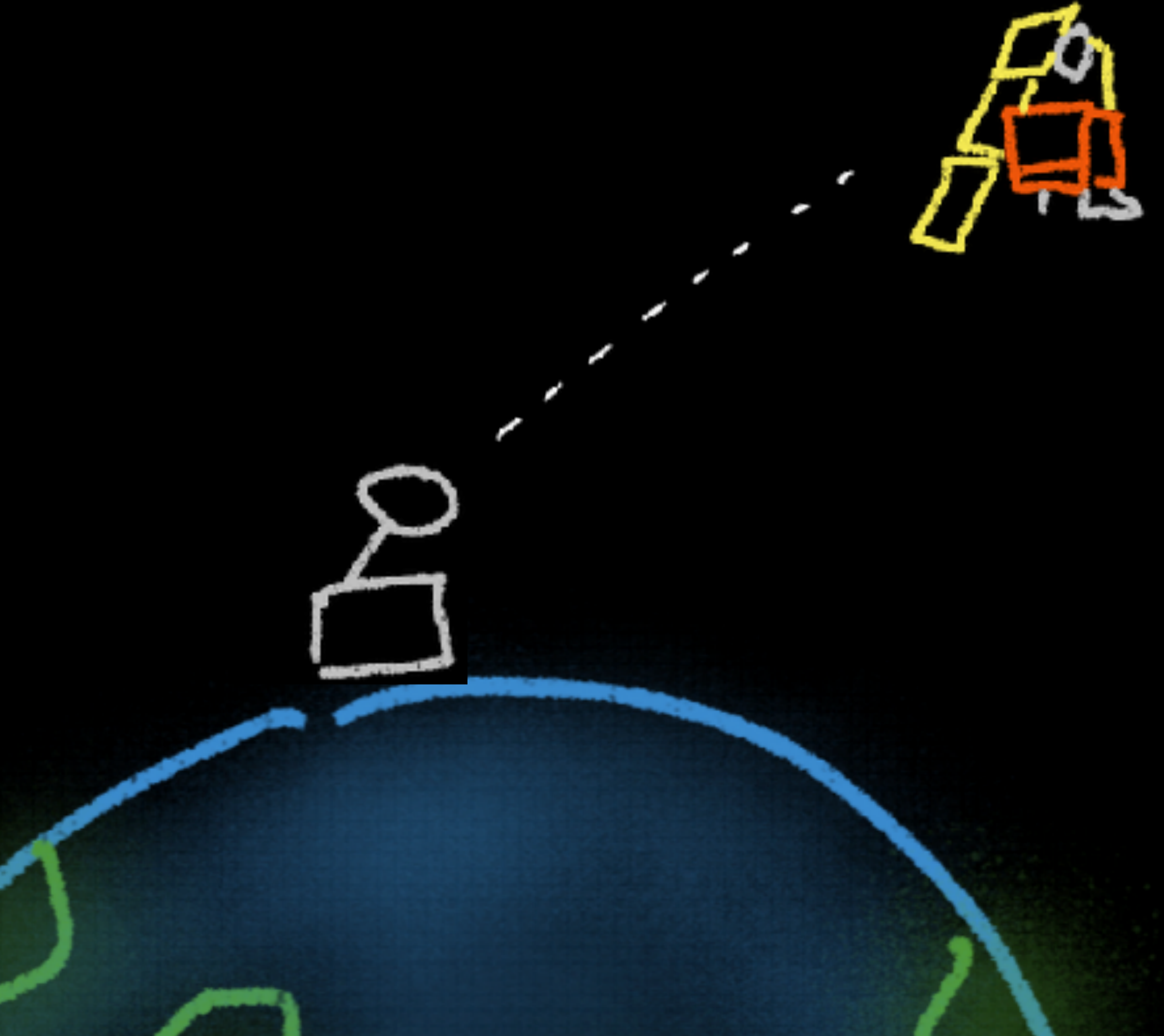
Domain

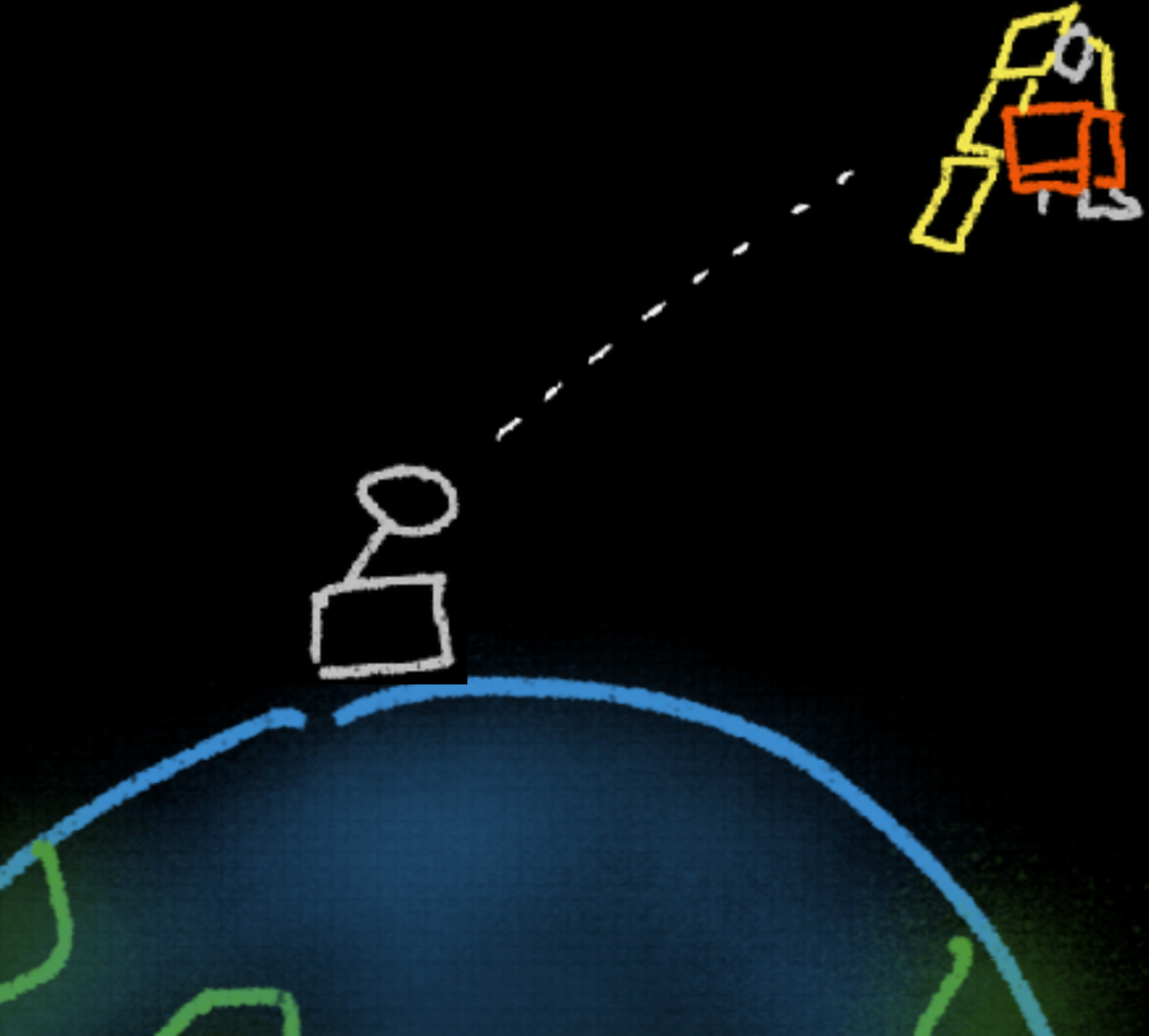



Courtesy NASA/
JPL-Caltech

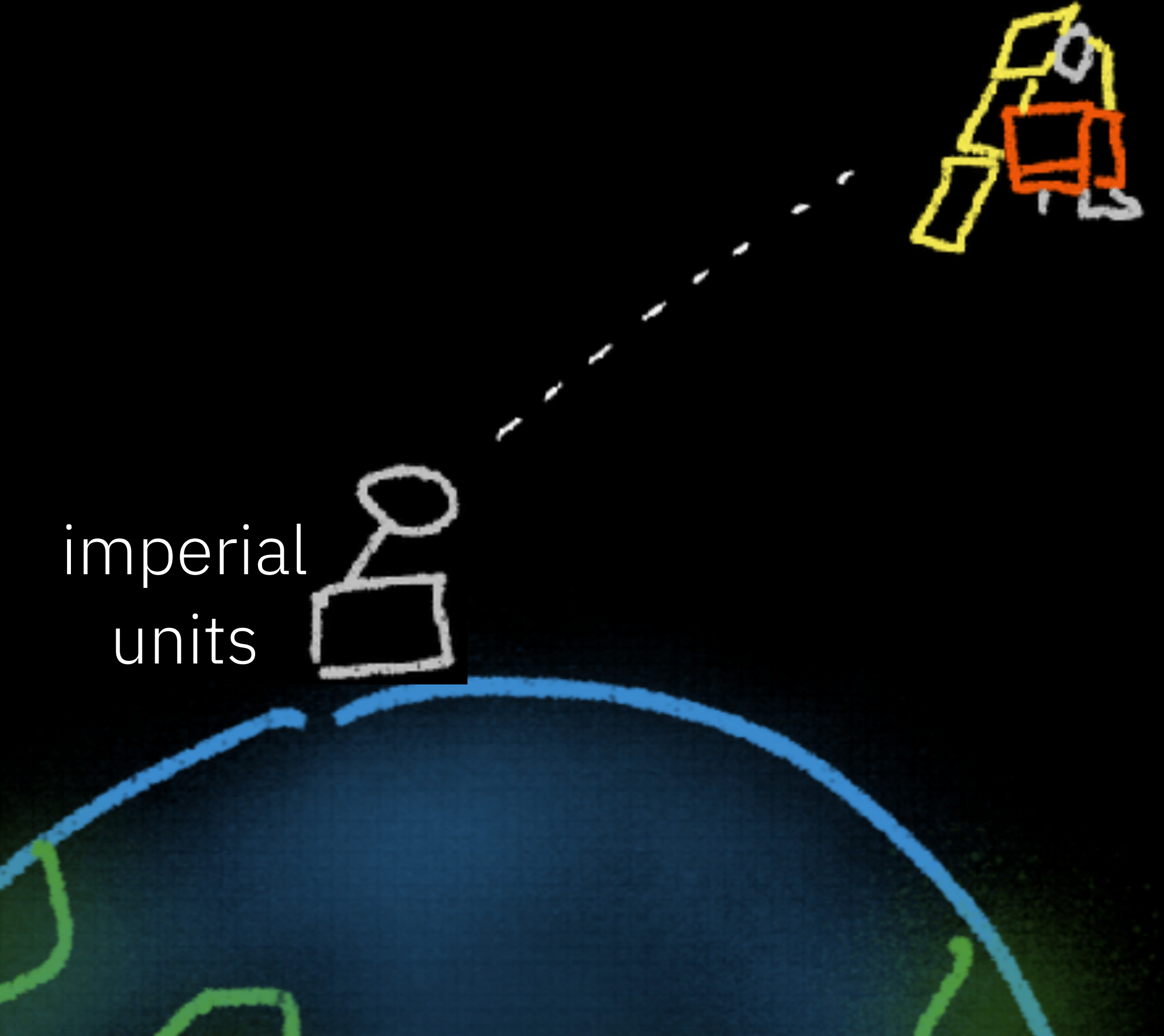


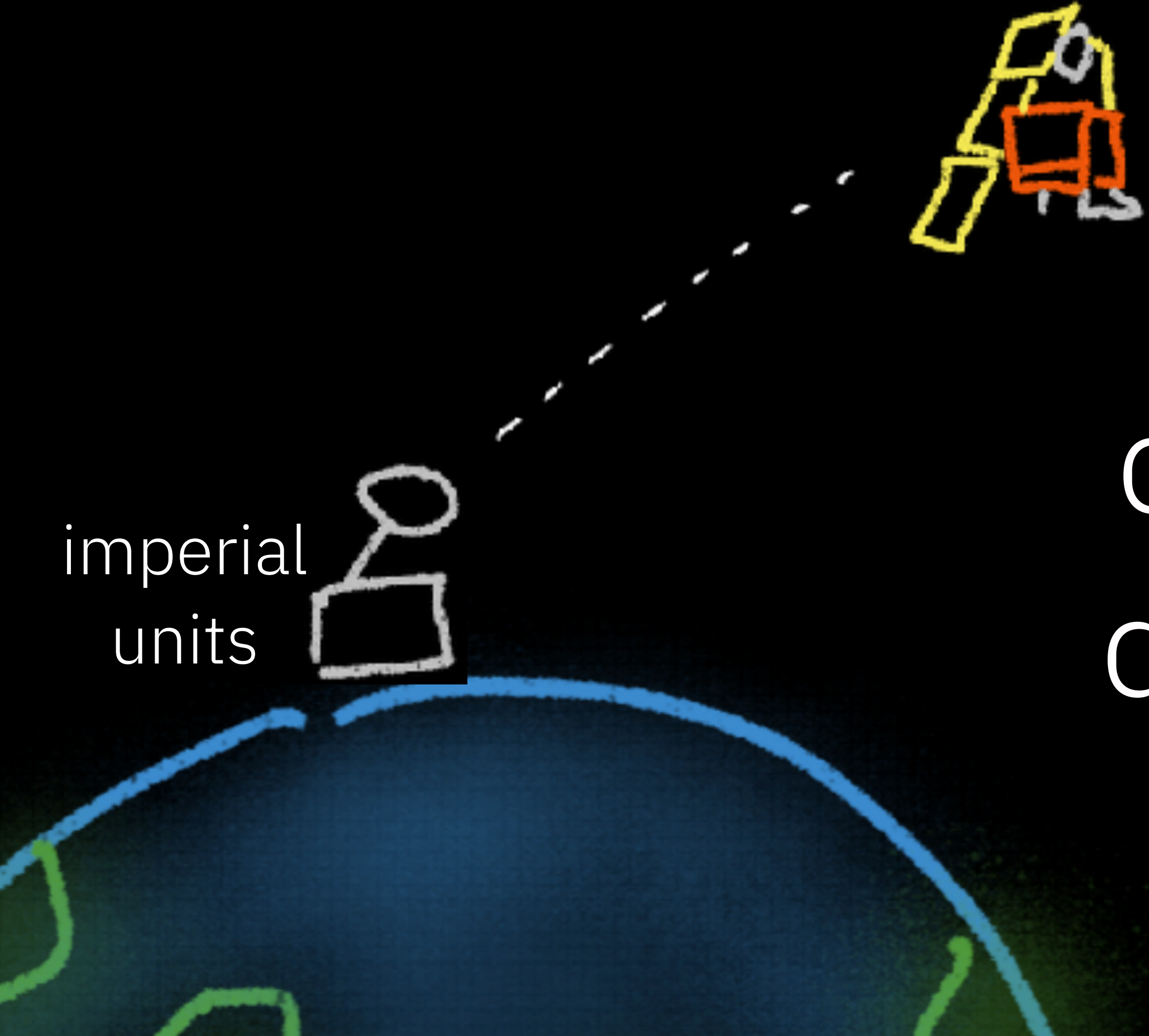






metric units


metric units
imperial units



distributing
did not help

microservices **need**
consumer-driven contract tests

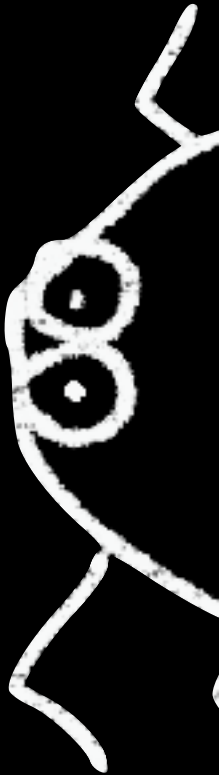
fail 7

the 'someday' automation

“our tests aren’t automated”

“we don’t know if
our code works”

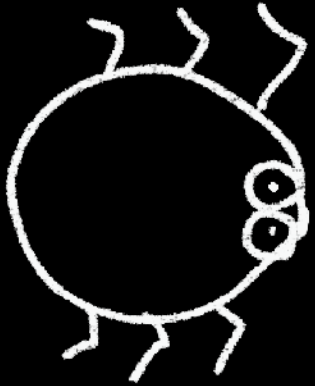
“we don’t know if our code works”



systems **will** behave in unexpected ways

dependency updates can change behaviour

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

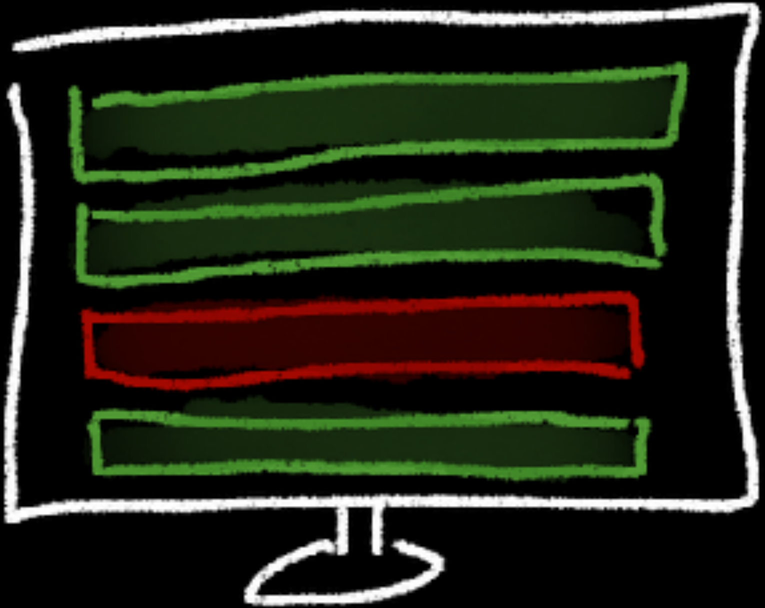


microservices **need** automated integration tests

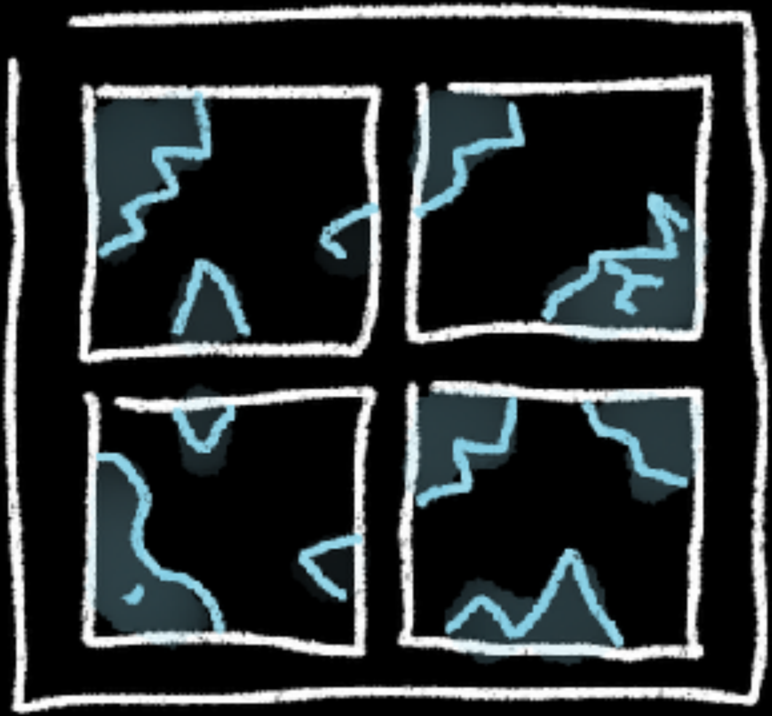
not a good CI/CD indicator

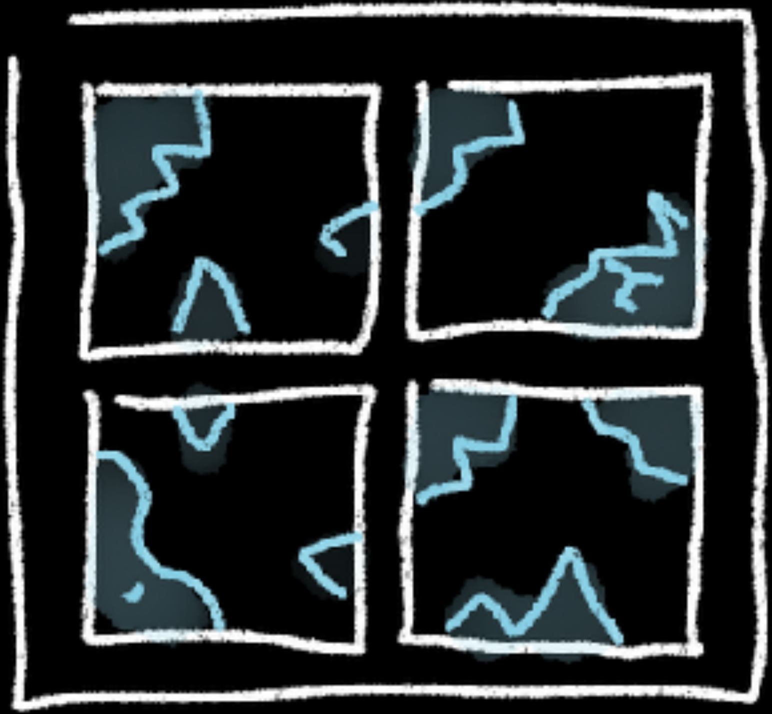
a good CI/CD indicator

"we don't know when the build is broken"



a good build radiator





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

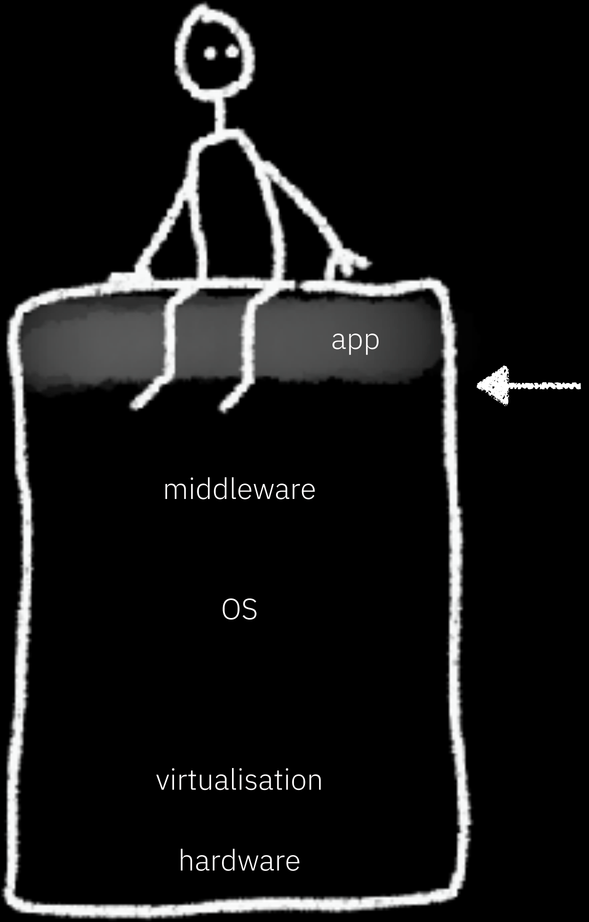
fail 8

microservices ops mayhem

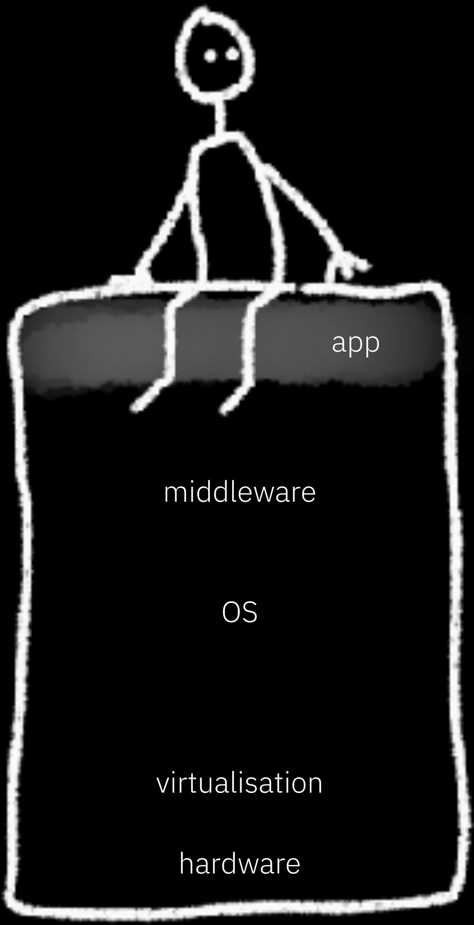
@holly_cummins

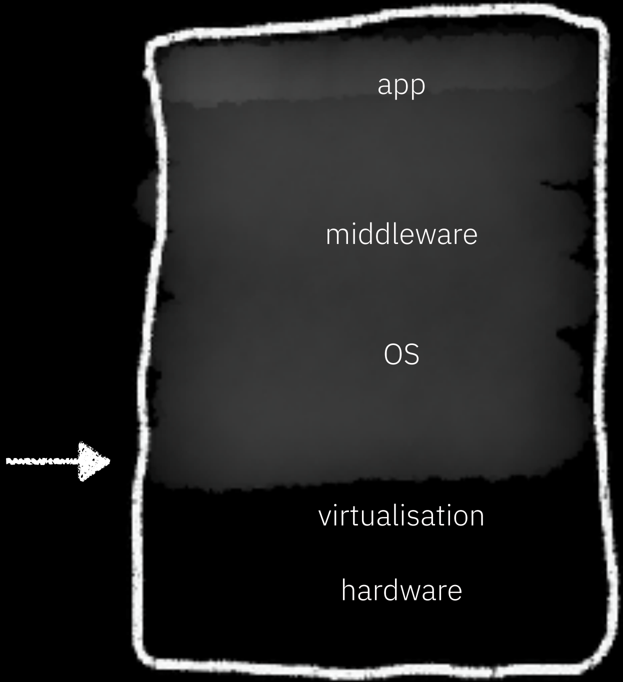
security

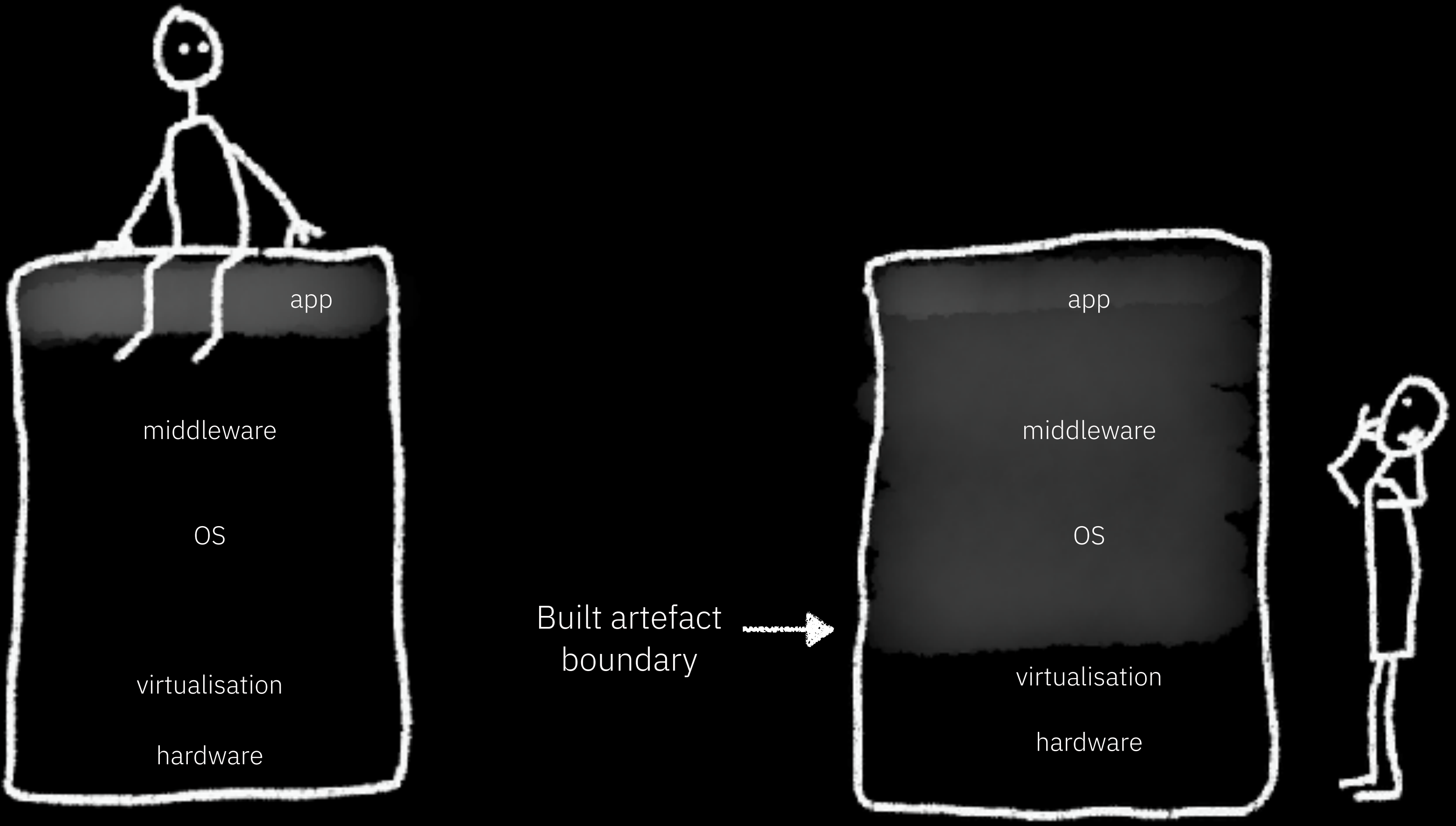
@holly_cummins



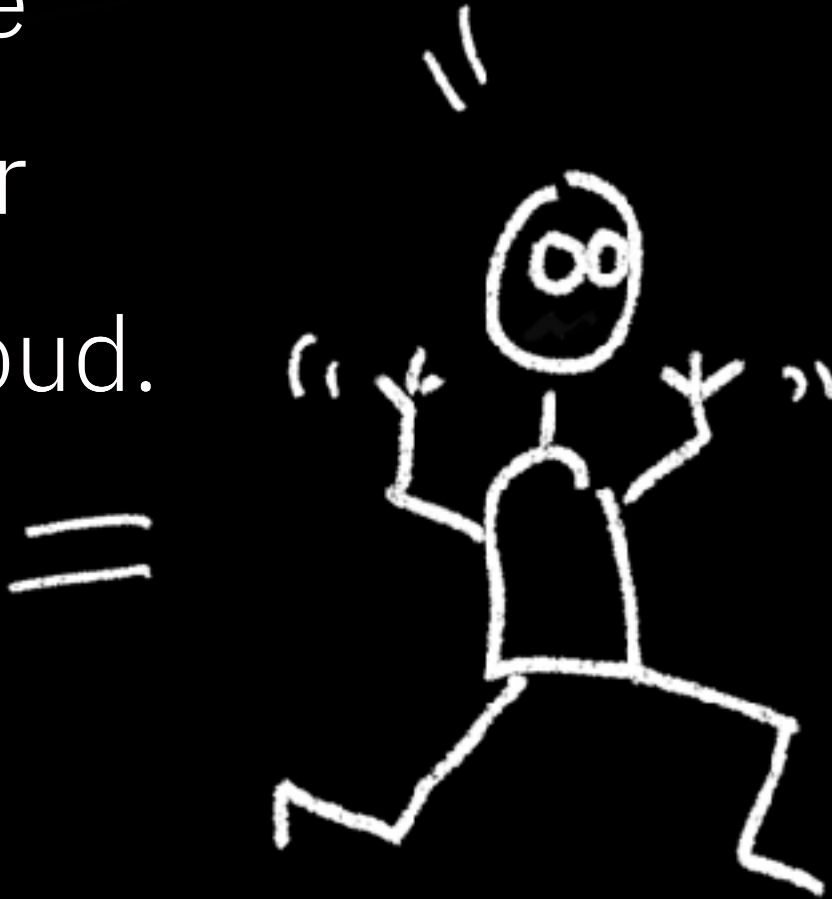
Built artefact boundary


app
middleware
OS
virtualisation
hardware


Built artefact boundary
app
middleware
OS
virtualisation
hardware

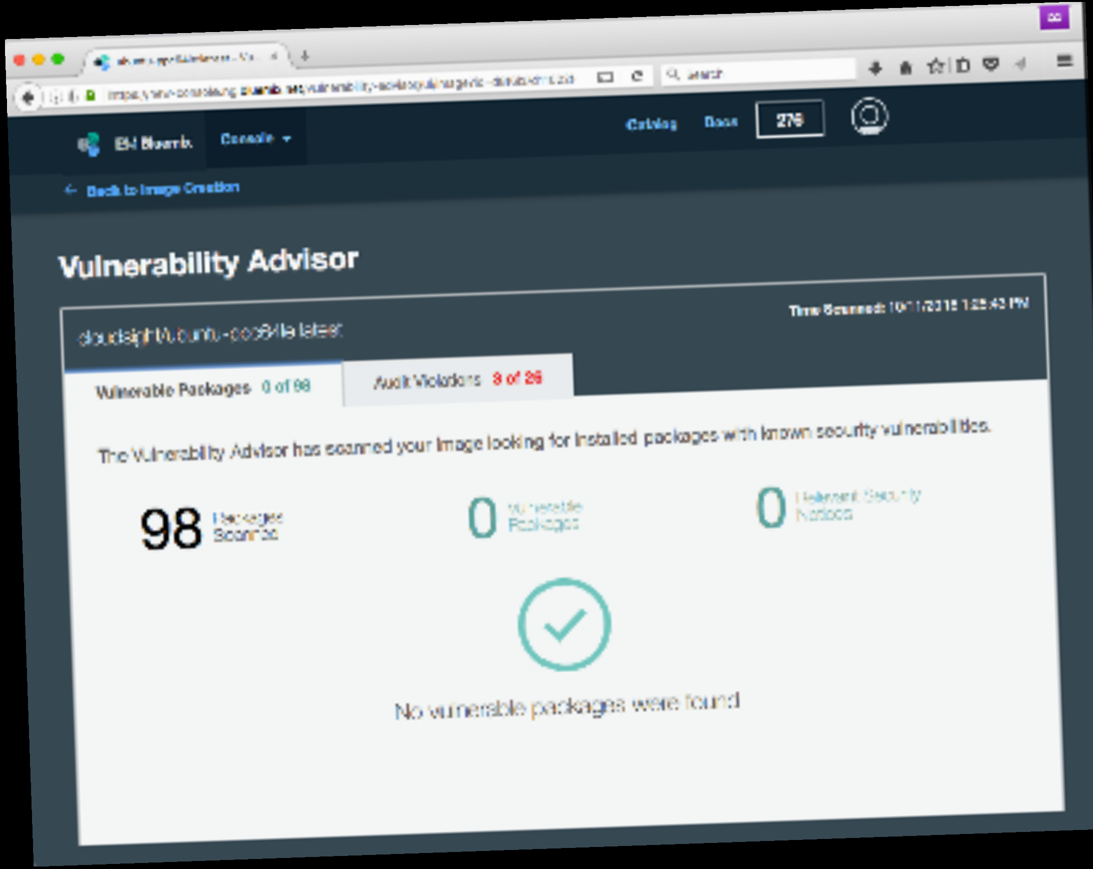

app
middleware
OS
virtualisation
hardware
Built artefact boundary
app
middleware
OS
virtualisation
hardware

Developers are responsible for security in the cloud.



build security

in everywhere



Vulnerability Advisor

Time Scanned: 10/11/2015 1:25:43 PM

cloudsight/ubuntu-cocktails:latest

Vulnerable Packages 0 of 98

Audit Violations 3 of 26

The Vulnerability Advisor has scanned your image looking for installed packages with known security vulnerabilities.

98 Packages Scanned

0 Vulnerable Packages

0 Relevant Security Notices

No vulnerable packages were found

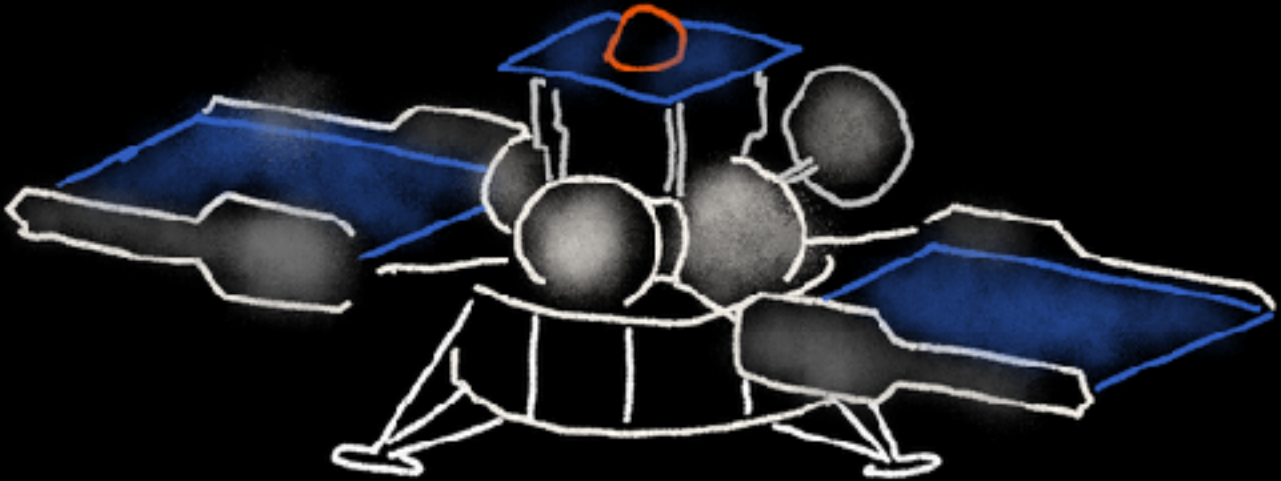
@holly_cummins

make

releases

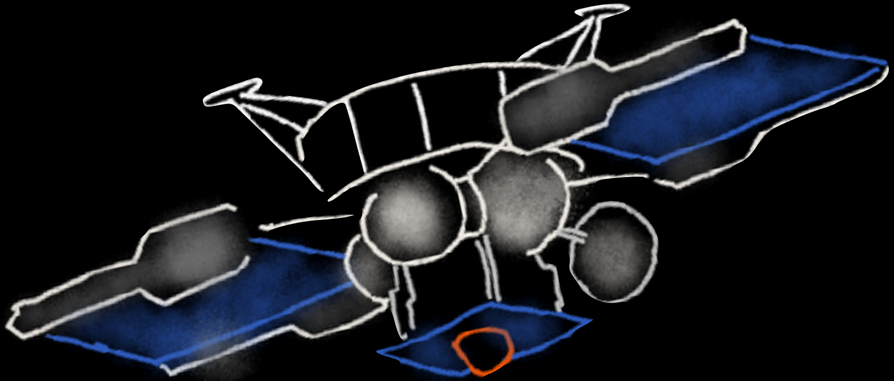
deeply **boring**

how to brick a spaceprobe



Phobos 1

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

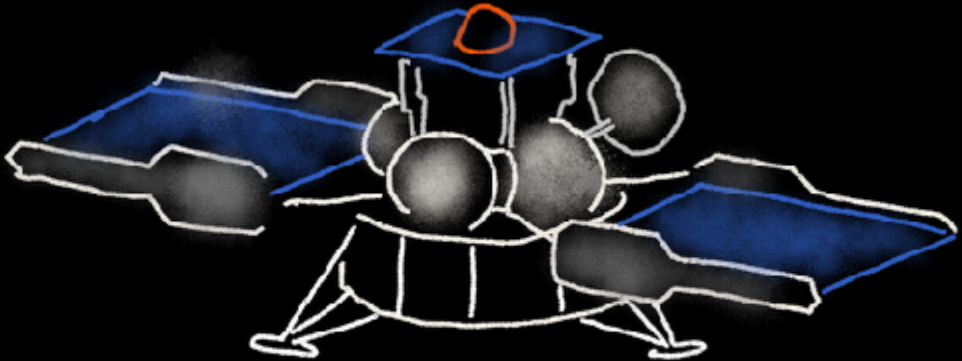


SRE

site reliability engineering

observability

recoverability



unrecoverable

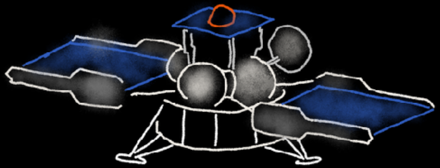
back in ms
no data loss

bricked

manual
intervention

fast, but
data lost

handoffs



handoffs bad
automation good

fail 9

microservices envy

containers are a good base

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

containers are a good base

“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.”

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

ways to succeed at cloud native

devops

be clear on what you're trying to achieve

align business and IT

collaborate with experts

co-creation is awesome

optimise for feedback

IBM®

@holly_cummins