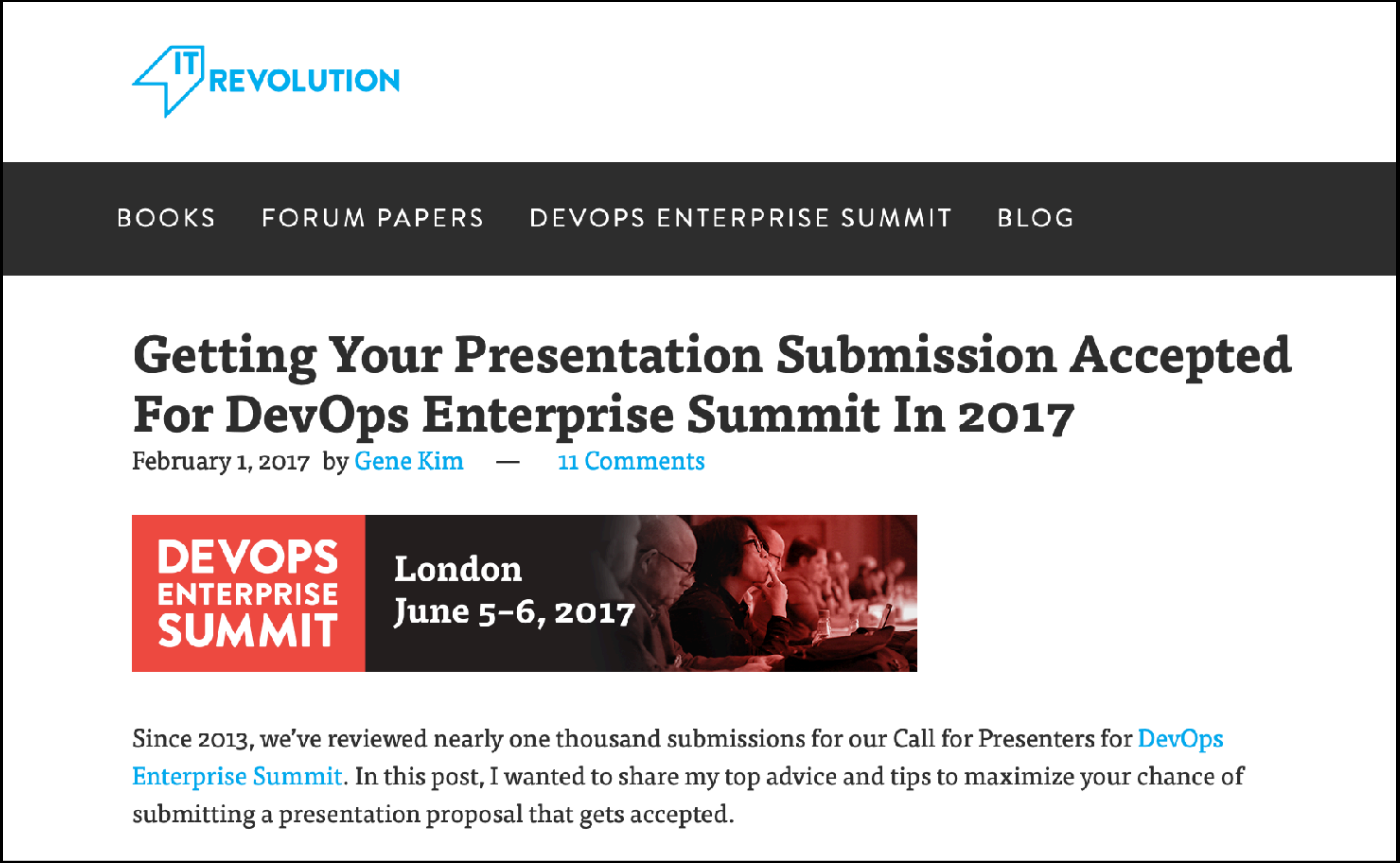
Tales from the DevOps Transformation Trenches

yes, you (still) need to start with culture, not containers

Holly Cummins
IBM Garage
@holly_cummins

IBM

Austin
Copenhagen
Dubai
London
Madrid
Melbourne
Munich
New York
Nice
Raleigh
San Francisco
São Paulo
Singapore
Tokyo
Toronto



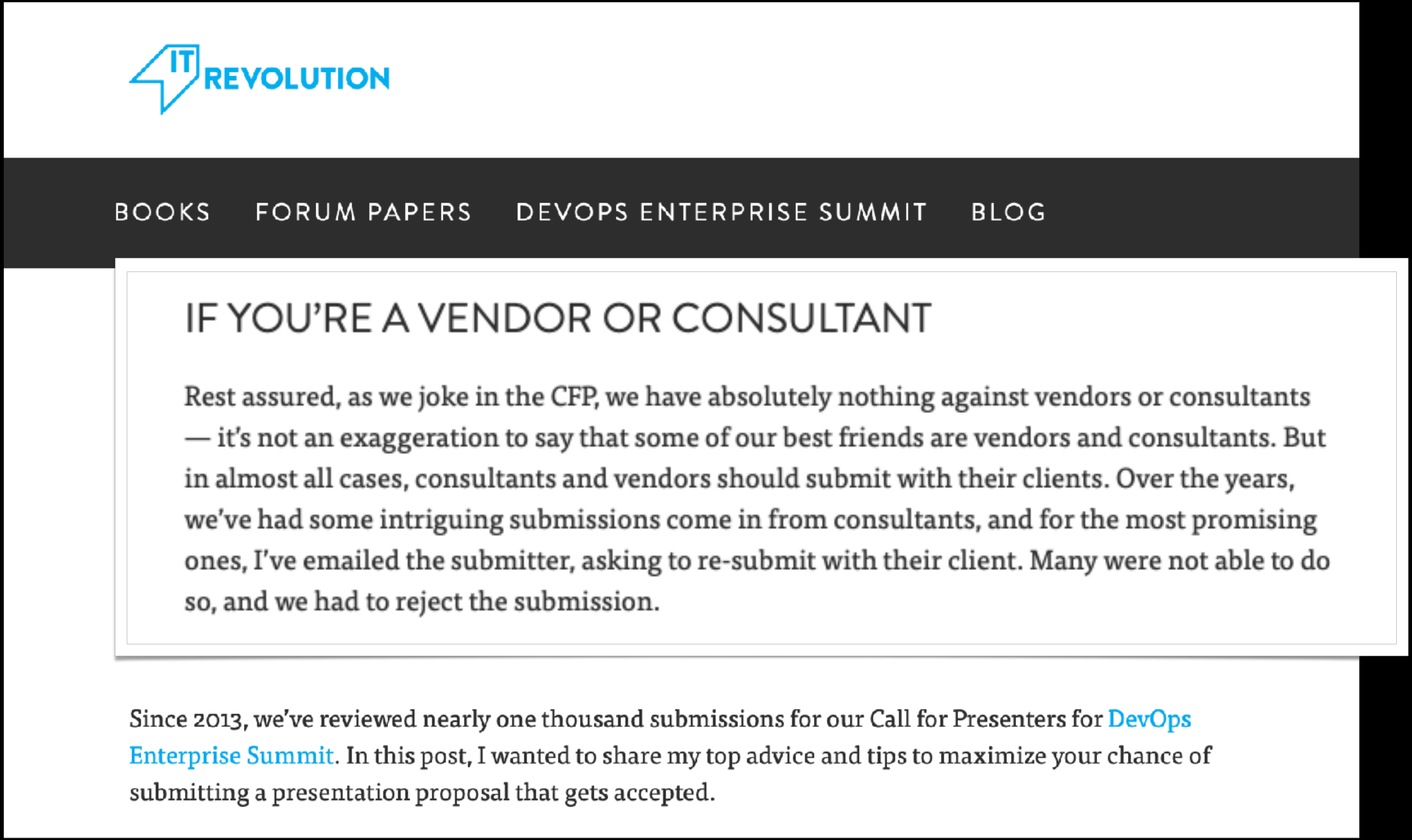
IT REVOLUTION

BOOKS FORUM PAPERS DEVOPS ENTERPRISE SUMMIT BLOG

Getting Your Presentation Submission Accepted For DevOps Enterprise Summit In 2017

February 1, 2017 by Gene Kim — 11 Comments

Since 2013, we've reviewed nearly one thousand submissions for our Call for Presenters for DevOps Enterprise Summit. In this post, I wanted to share my top advice and tips to maximize your chance of submitting a presentation proposal that gets accepted.

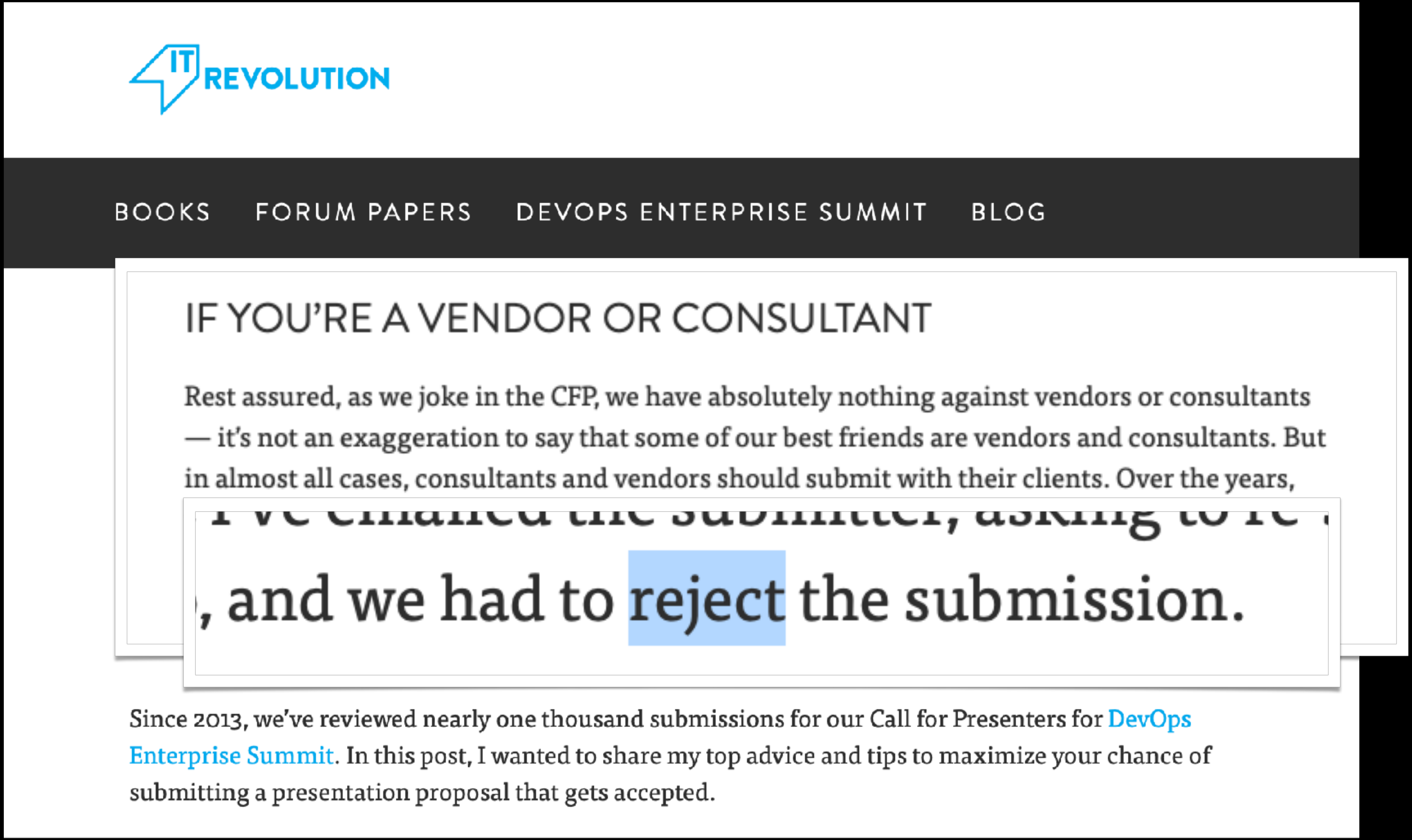

IT REVOLUTION

BOOKS FORUM PAPERS DEVOPS ENTERPRISE SUMMIT BLOG

IF YOU'RE A VENDOR OR CONSULTANT

Rest assured, as we joke in the CFP, we have absolutely nothing against vendors or consultants — it's not an exaggeration to say that some of our best friends are vendors and consultants. But in almost all cases, consultants and vendors should submit with their clients. Over the years, we've had some intriguing submissions come in from consultants, and for the most promising ones, I've emailed the submitter, asking to re-submit with their client. Many were not able to do so, and we had to reject the submission.

Since 2013, we've reviewed nearly one thousand submissions for our Call for Presenters for DevOps Enterprise Summit. In this post, I wanted to share my top advice and tips to maximize your chance of submitting a presentation proposal that gets accepted.



IT REVOLUTION

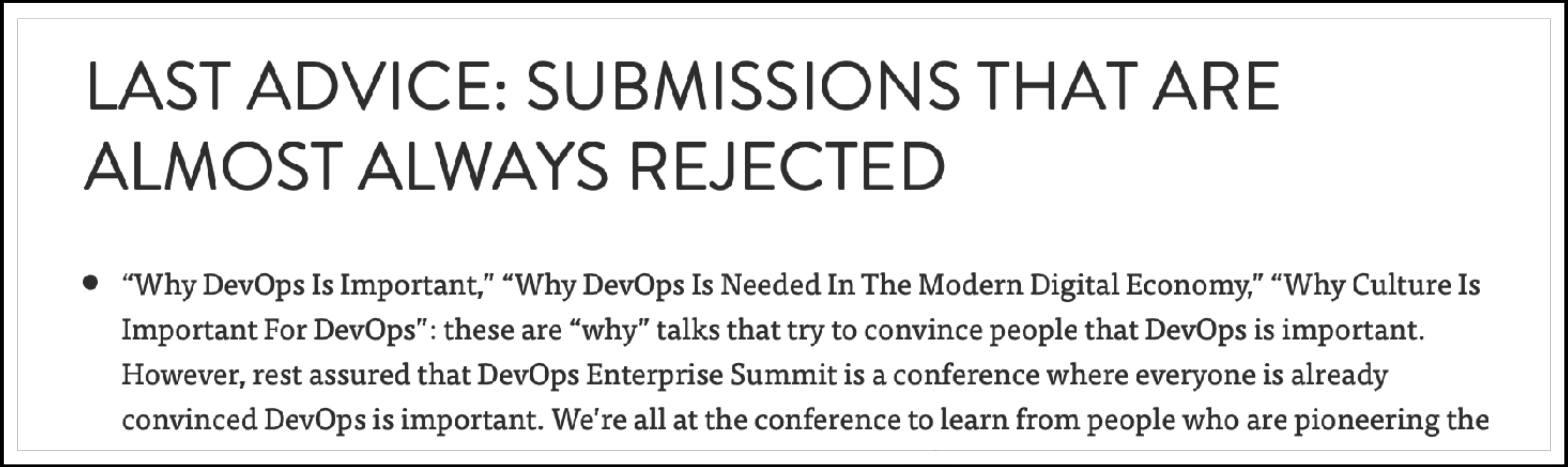
BOOKS FORUM PAPERS DEVOPS ENTERPRISE SUMMIT BLOG

IF YOU'RE A VENDOR OR CONSULTANT

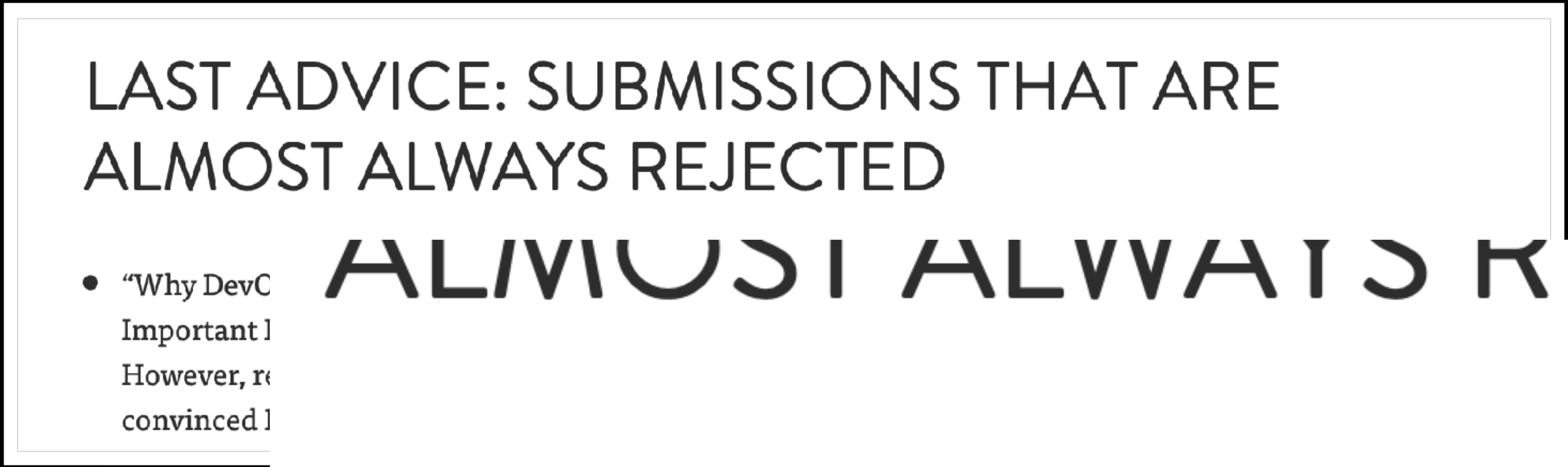
Rest assured, as we joke in the CFP, we have absolutely nothing against vendors or consultants — it's not an exaggeration to say that some of our best friends are vendors and consultants. But in almost all cases, consultants and vendors should submit with their clients. Over the years,

, and we had to reject the submission.

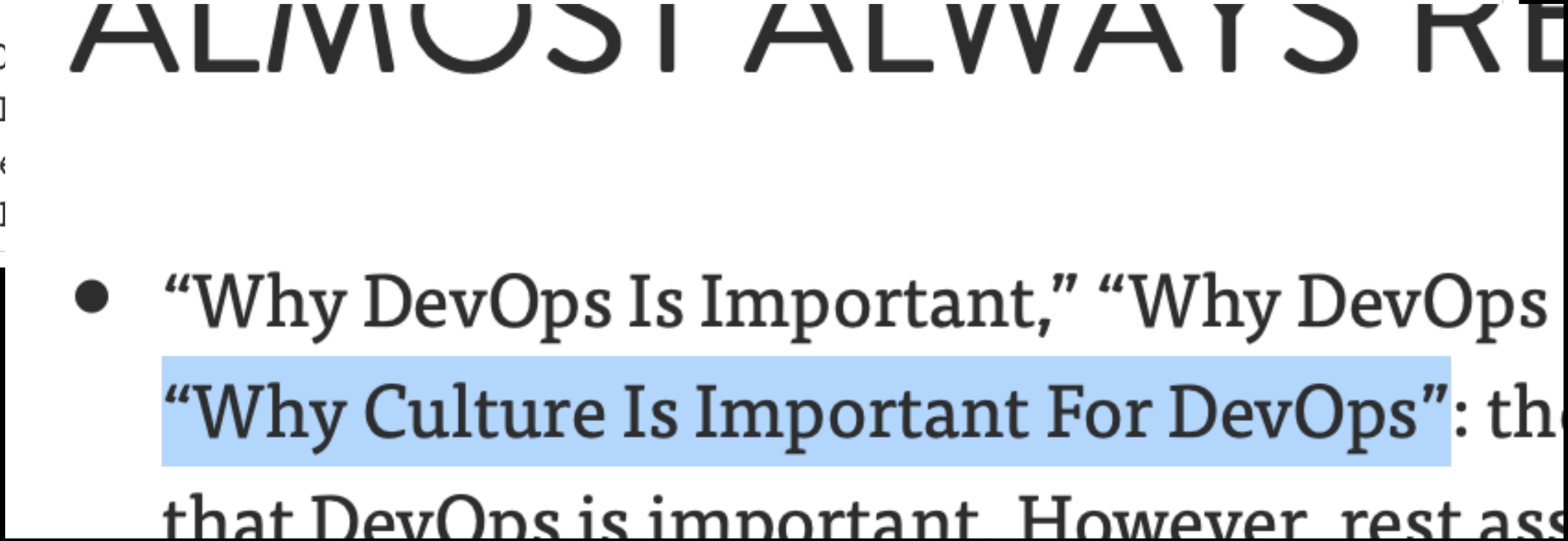
Since 2013, we've reviewed nearly one thousand submissions for our Call for Presenters for DevOps Enterprise Summit. In this post, I wanted to share my top advice and tips to maximize your chance of submitting a presentation proposal that gets accepted.


LAST ADVICE: SUBMISSIONS THAT ARE ALMOST ALWAYS REJECTED

- "Why DevOps Is Important," "Why DevOps Is Needed In The Modern Digital Economy," "Why Culture Is Important For DevOps": these are "why" talks that try to convince people that DevOps is important. However, rest assured that DevOps Enterprise Summit is a conference where everyone is already convinced DevOps is important. We're all at the conference to learn from people who are pioneering the


LAST ADVICE: SUBMISSIONS THAT ARE ALMOST ALWAYS REJECTED

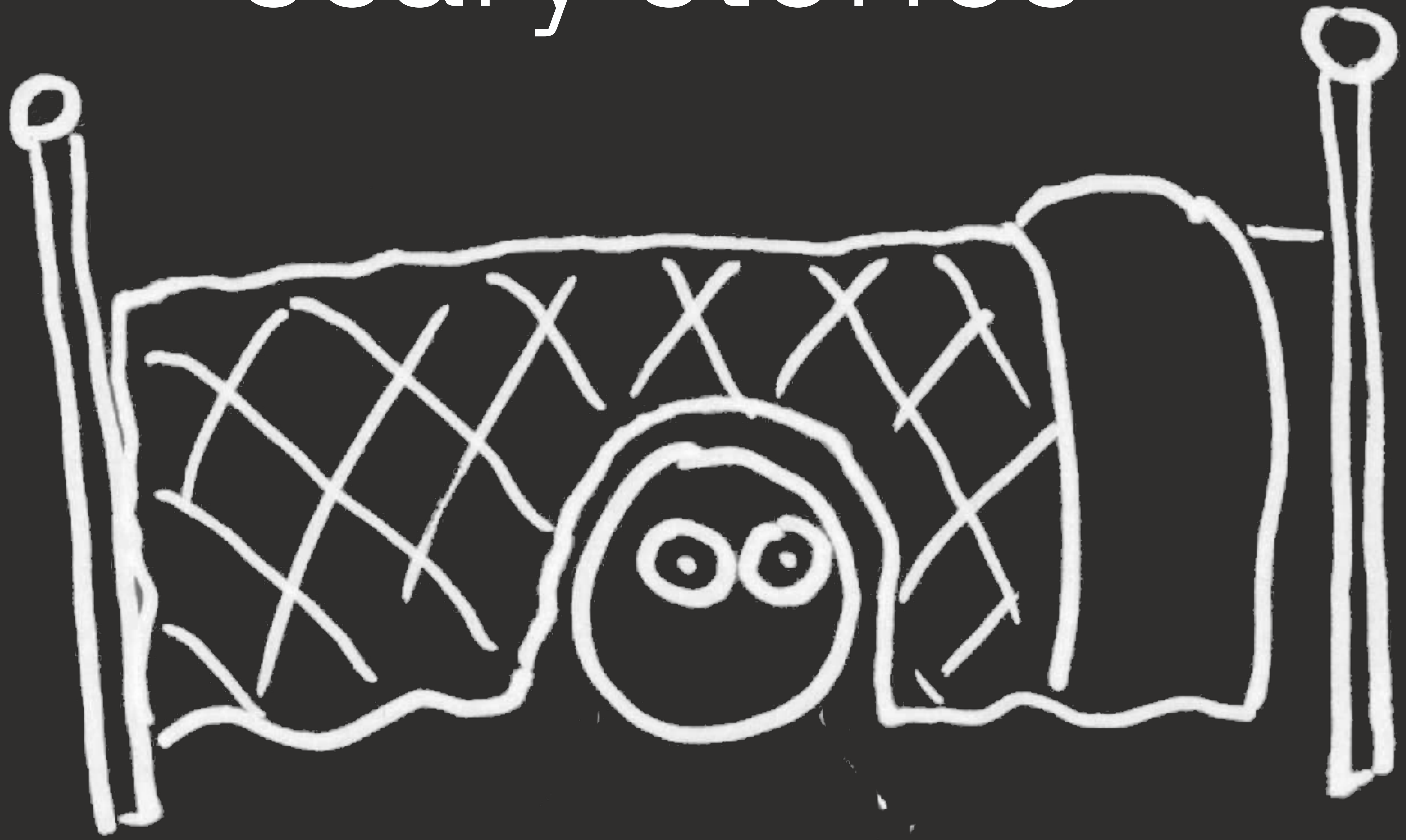
- "Why DevC Important I However, re convinced I


ALMOST ALWAYS RE

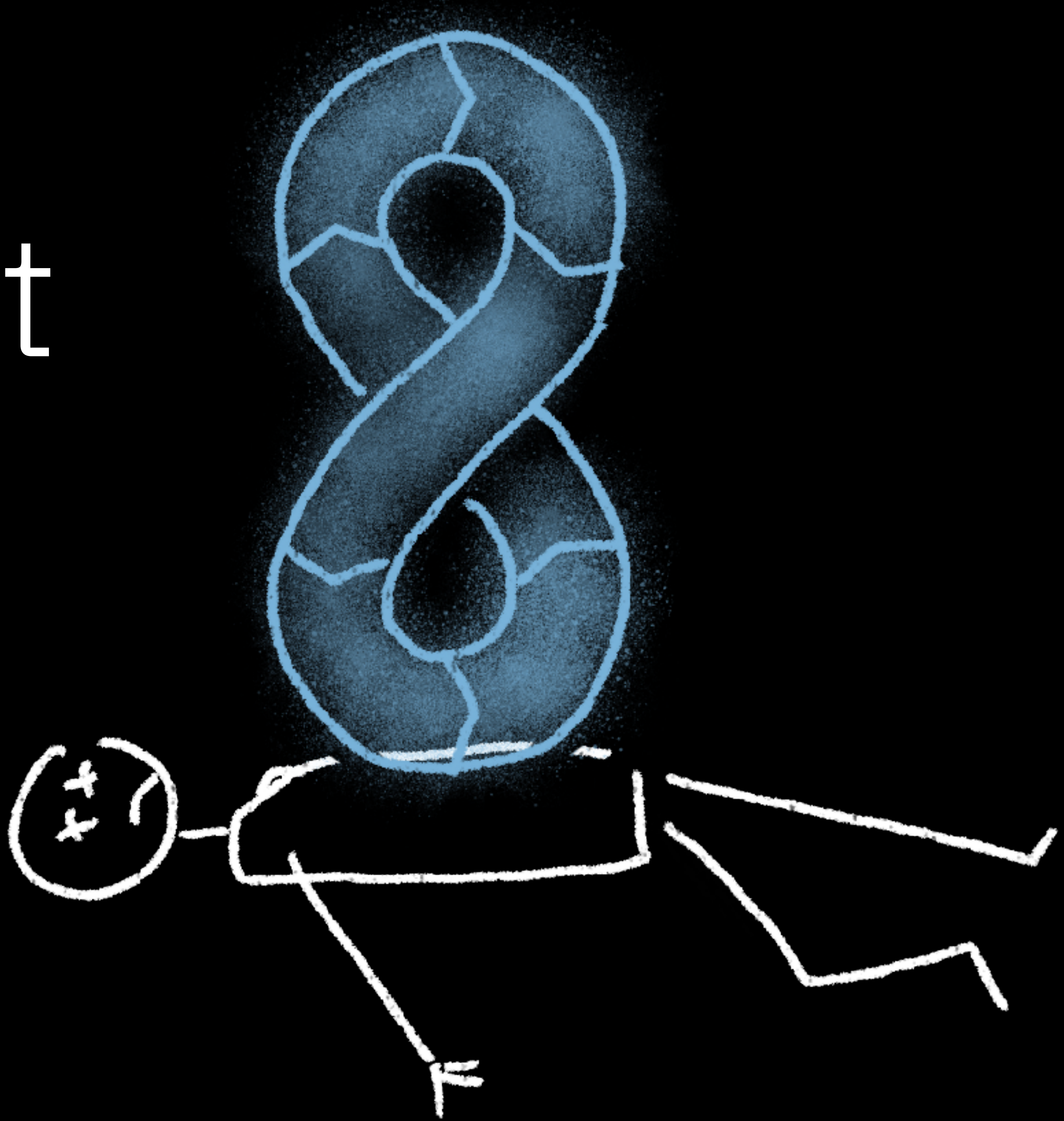
- "Why DevOps Is Important," "Why DevOps "Why Culture Is Important For DevOps": th that DevOps is important. However, rest ass hi.

i’m a consultant.

these are my scary stories



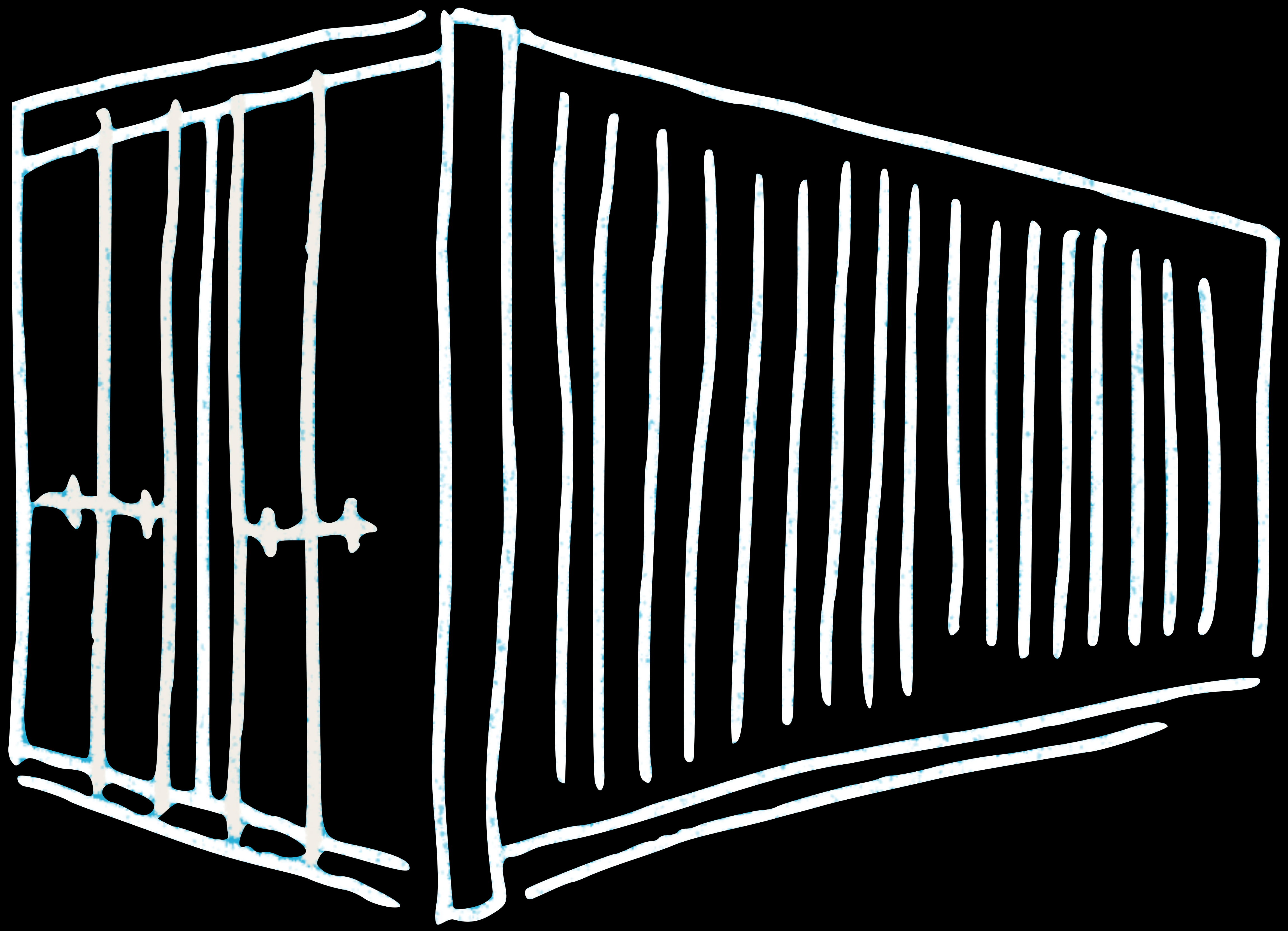
how to fail at devops



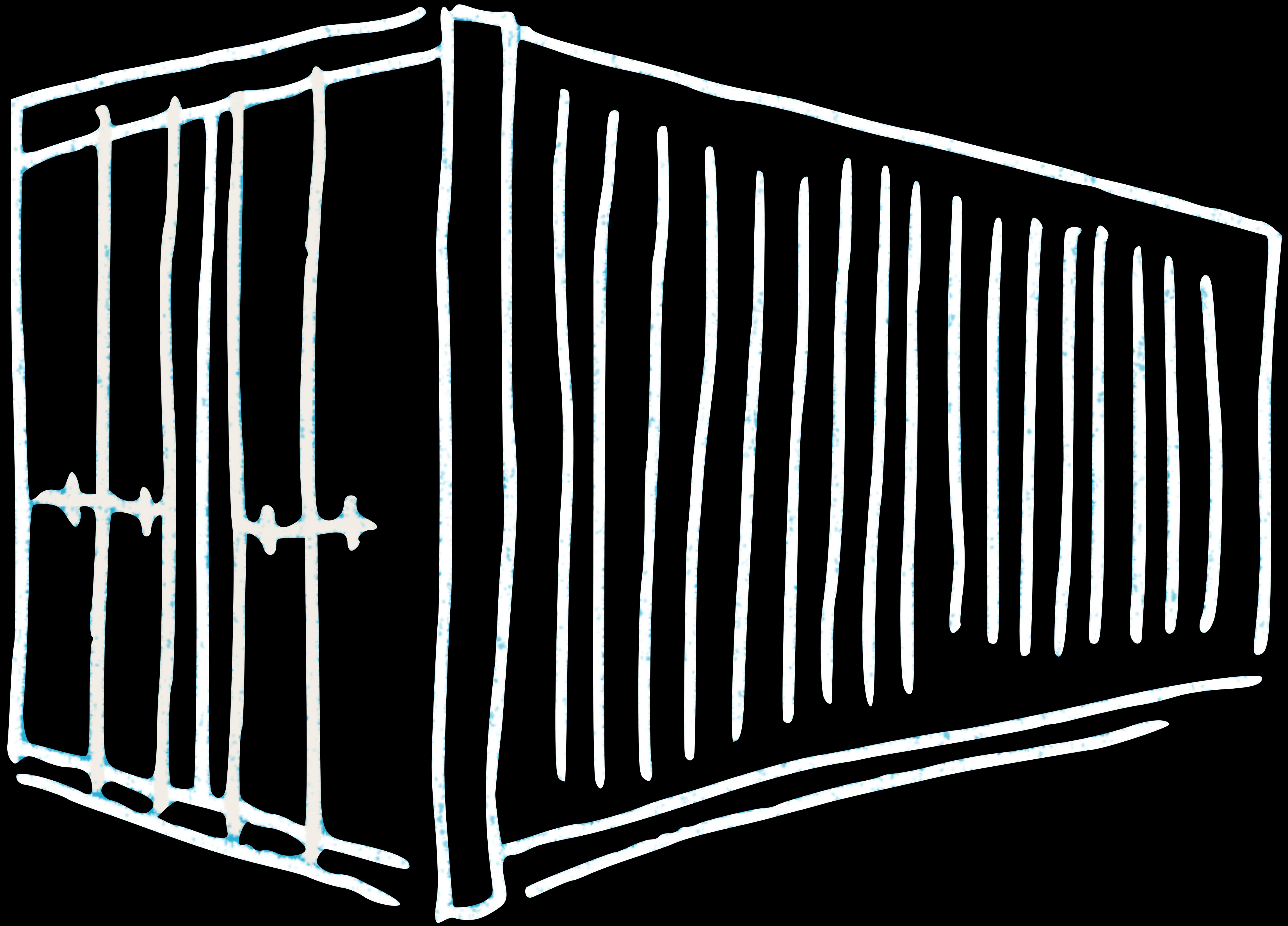
“this is our devops team”

“this is our devops team”

“… last year we called them the build team.”



containers will not fix your broken devops culture



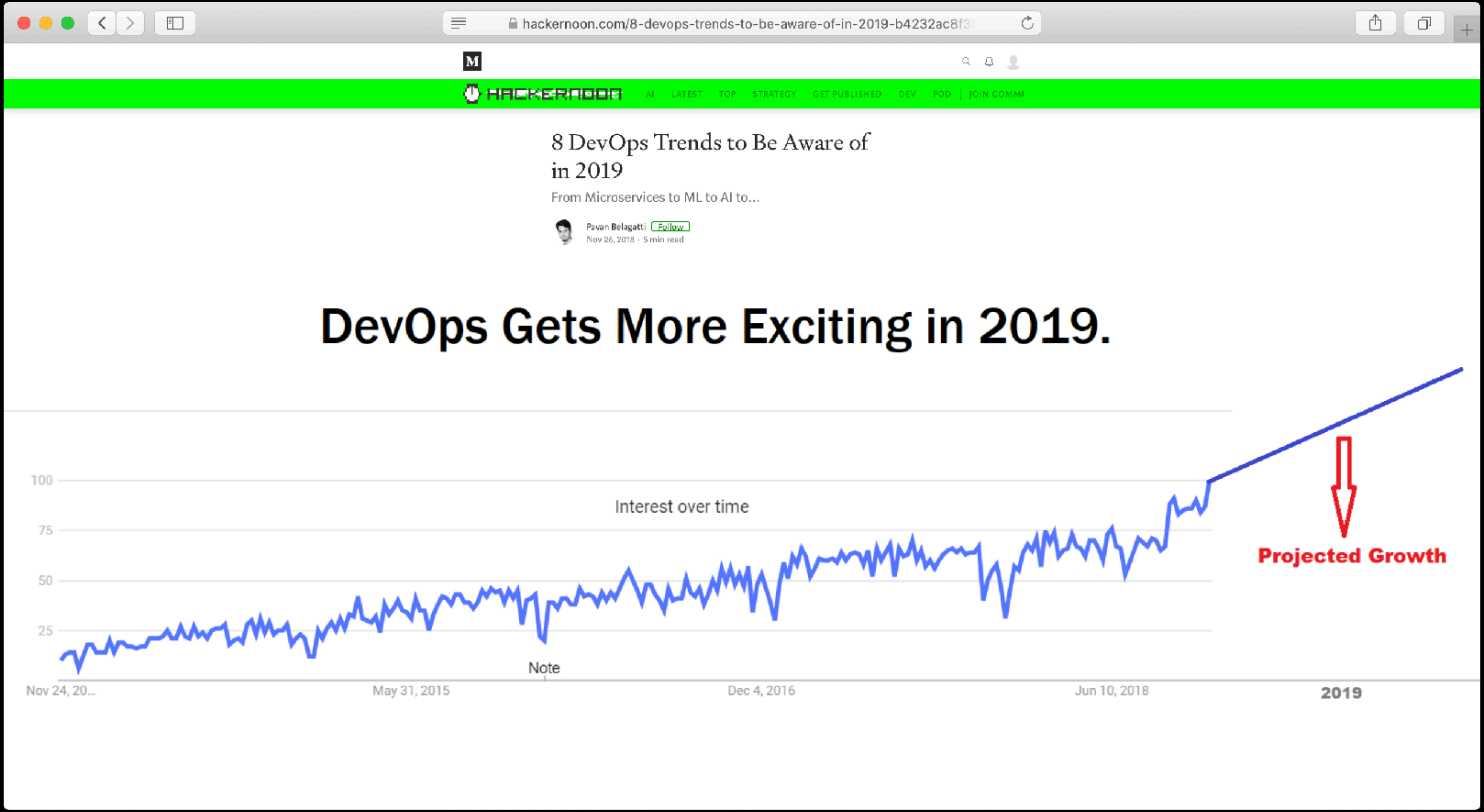
even kubernetes will not fix your broken devops culture “we’re going too slowly.

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.”


hackernoon.com/8-devops-trends-to-be-aware-of-in-2019-b4232ac8f3

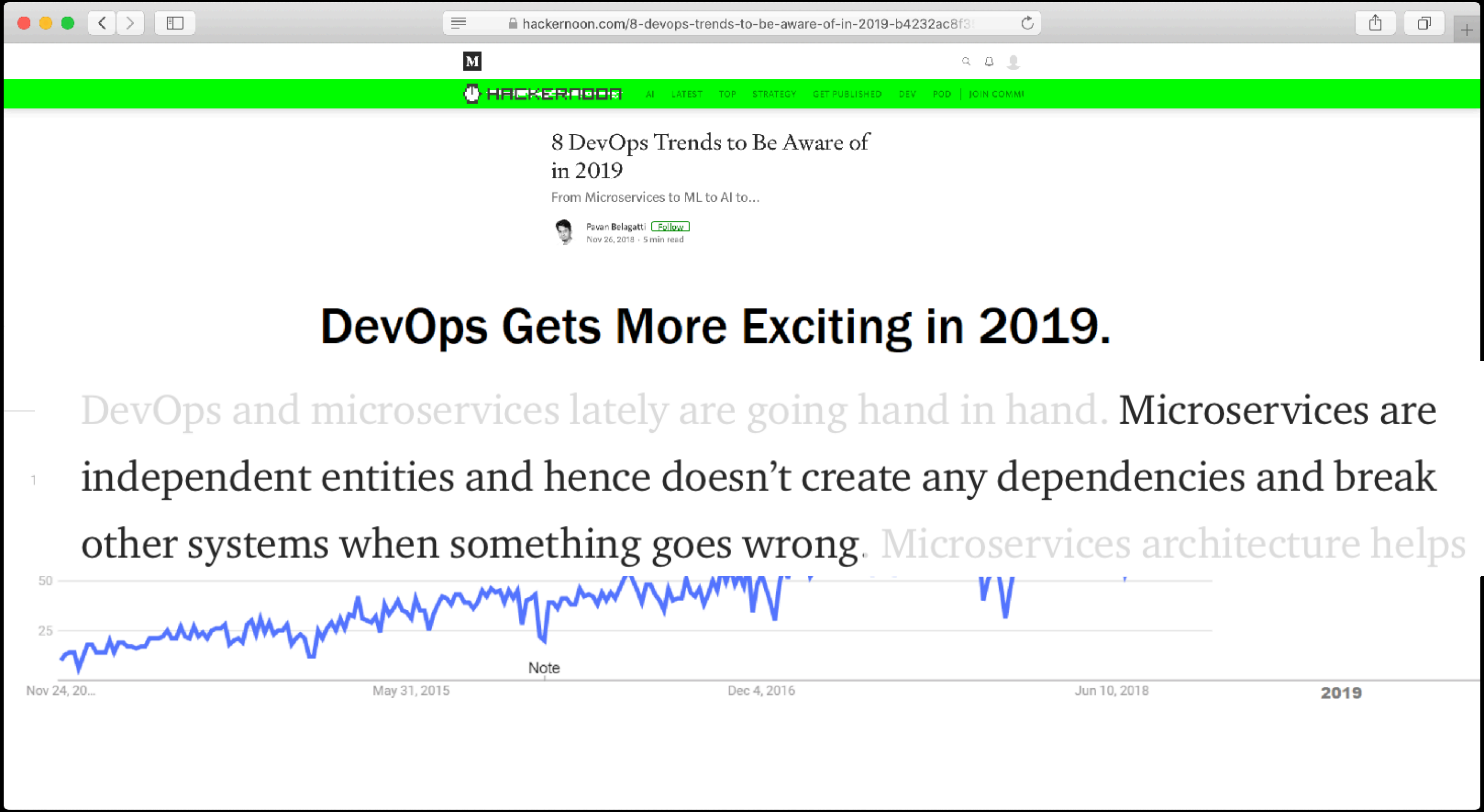
8 DevOps Trends to Be Aware of in 2019

From Microservices to ML to AI to...

Pavan Belagatti
Nov 26, 2018 · 5 min read

DevOps Gets More Exciting in 2019.

https://hackernoon.com/8-devops-trends-to-be-aware-of-in-2019-b4232ac8f351


hackernoon.com/8-devops-trends-to-be-aware-of-in-2019-b4232ac8f3

8 DevOps Trends to Be Aware of in 2019

From Microservices to ML to AI to...

Pavan Belagatti Follow
Nov 26, 2018 · 5 min read

DevOps Gets More Exciting in 2019.

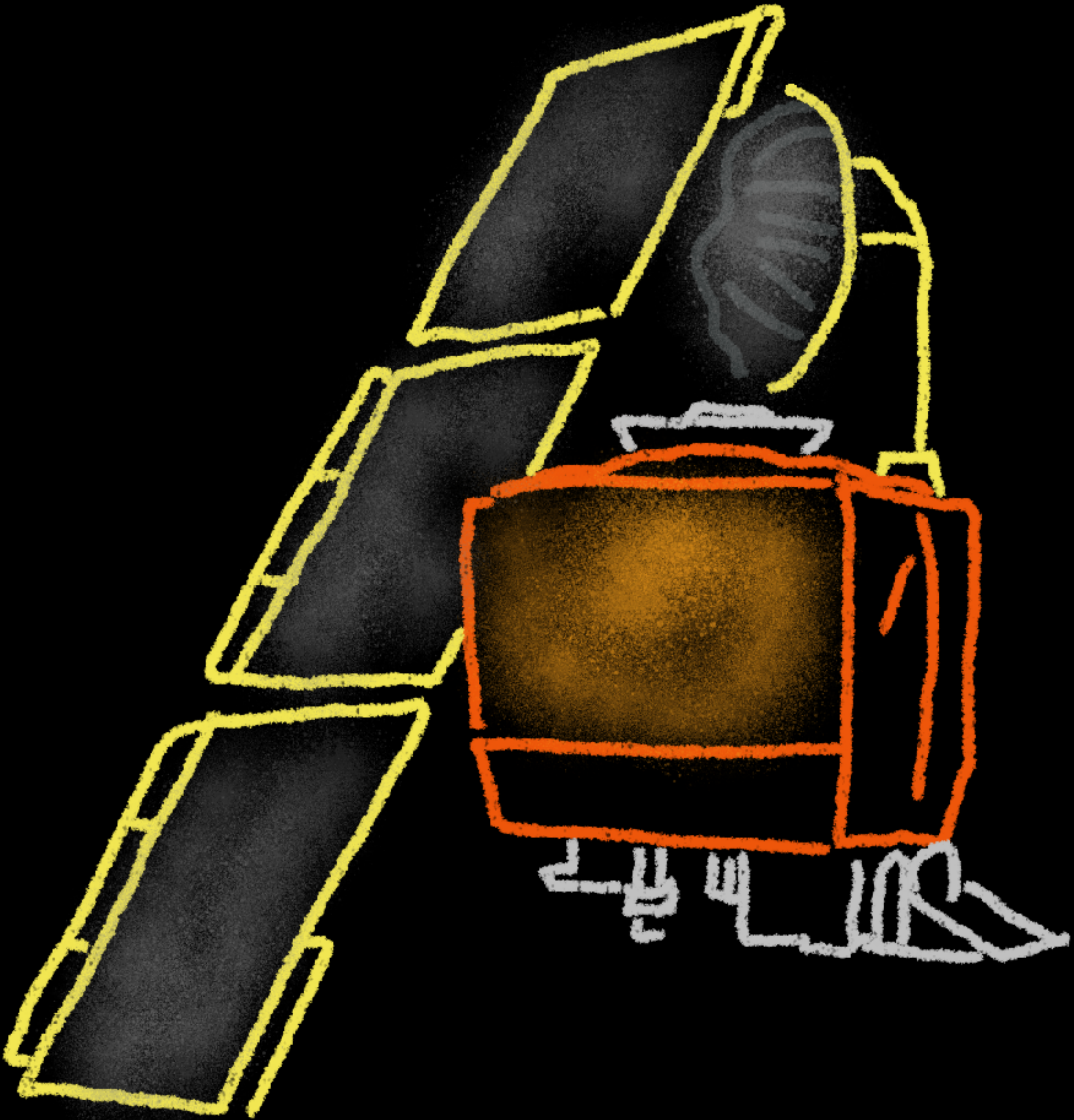
DevOps and microservices lately are going hand in hand. Microservices are independent entities and hence doesn't create any dependencies and break other systems when something goes wrong. Microservices architecture helps

https://hackernoon.com/8-devops-trends-to-be-aware-of-in-2019-b4232ac8f351 “every time we change code, something breaks” distributed monolith

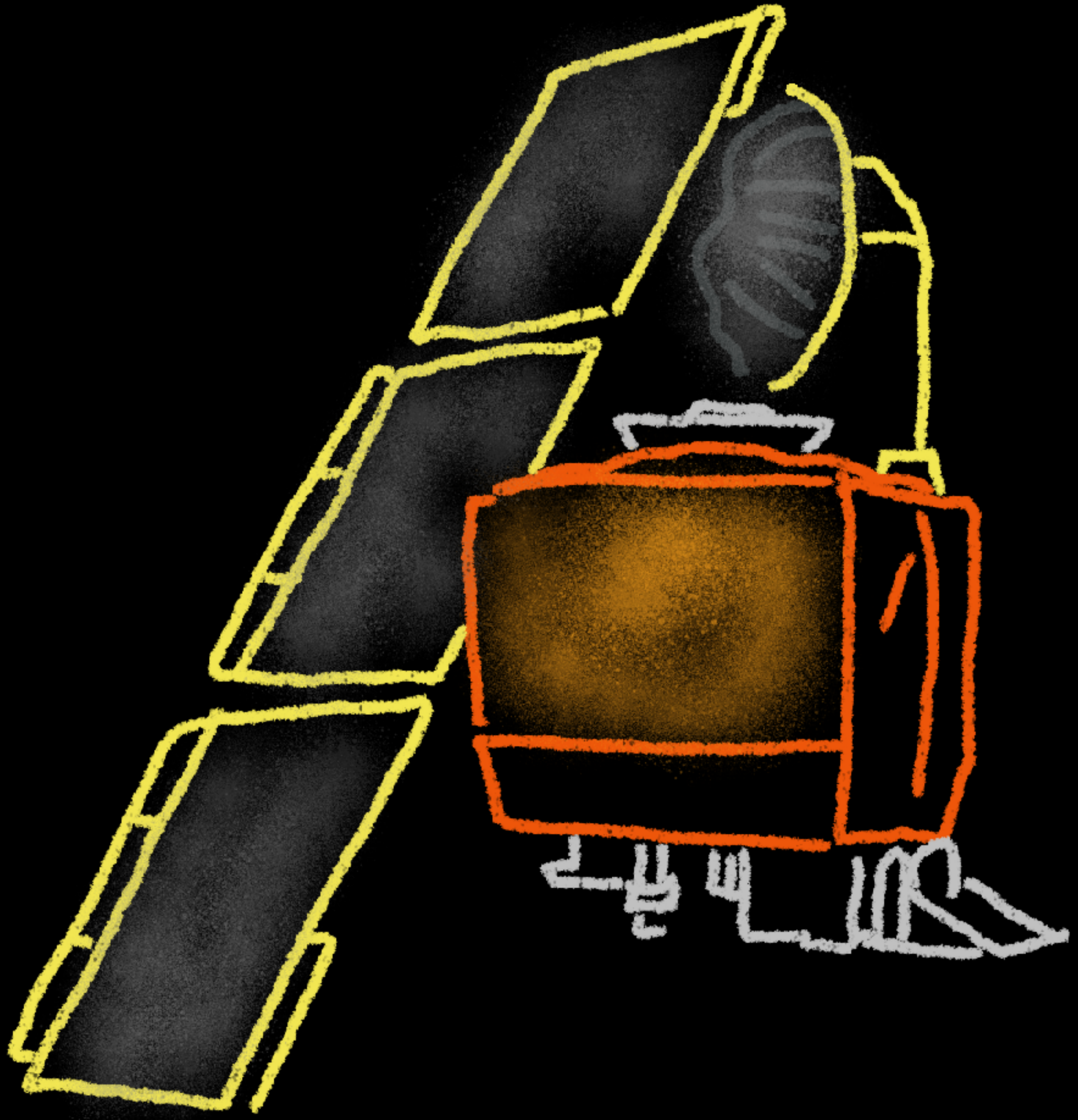
distributed monolith

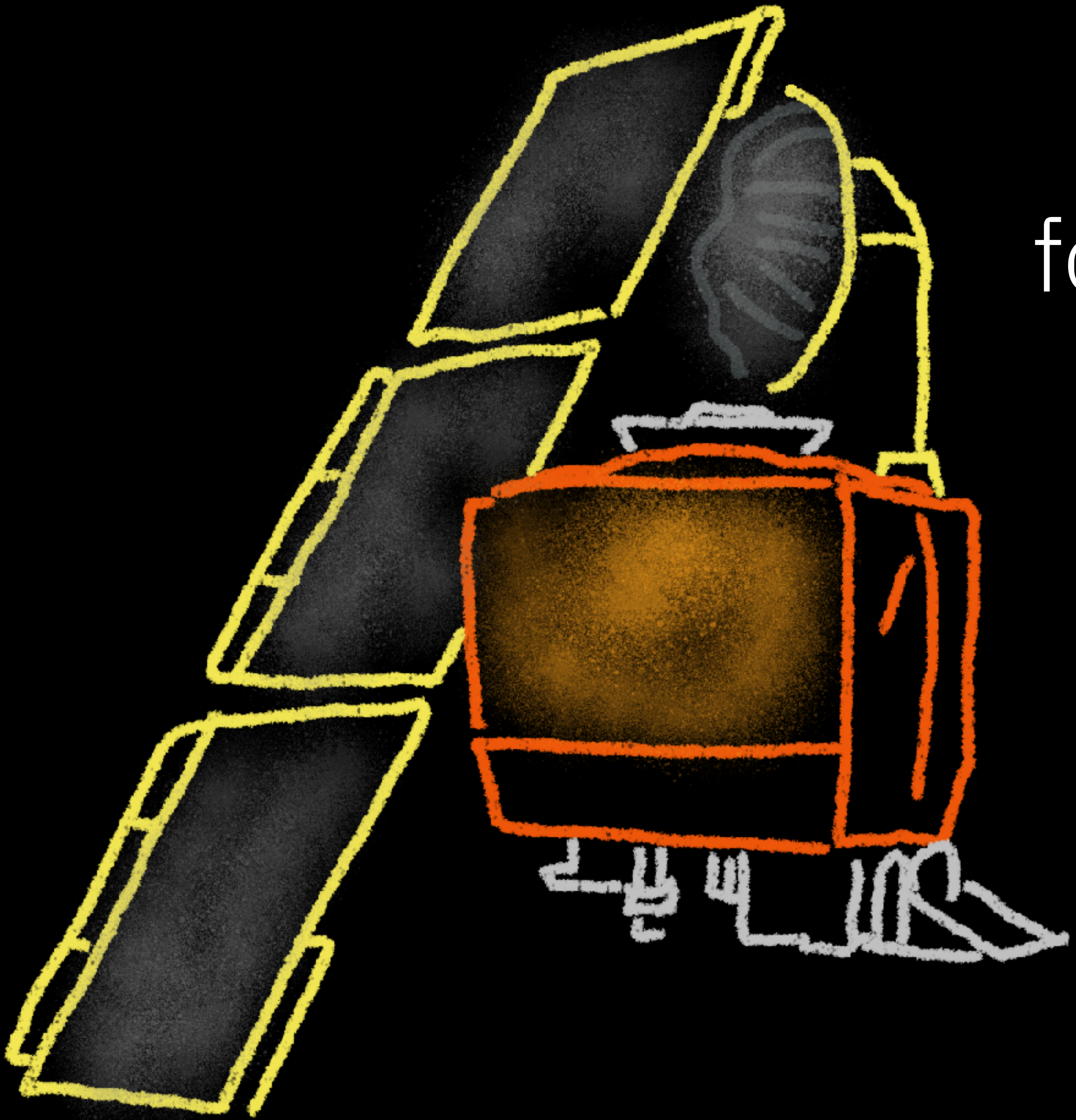
but without compile-time checking

just because a system runs across 6 containers doesn’t mean it’s decoupled



mars climate explorer

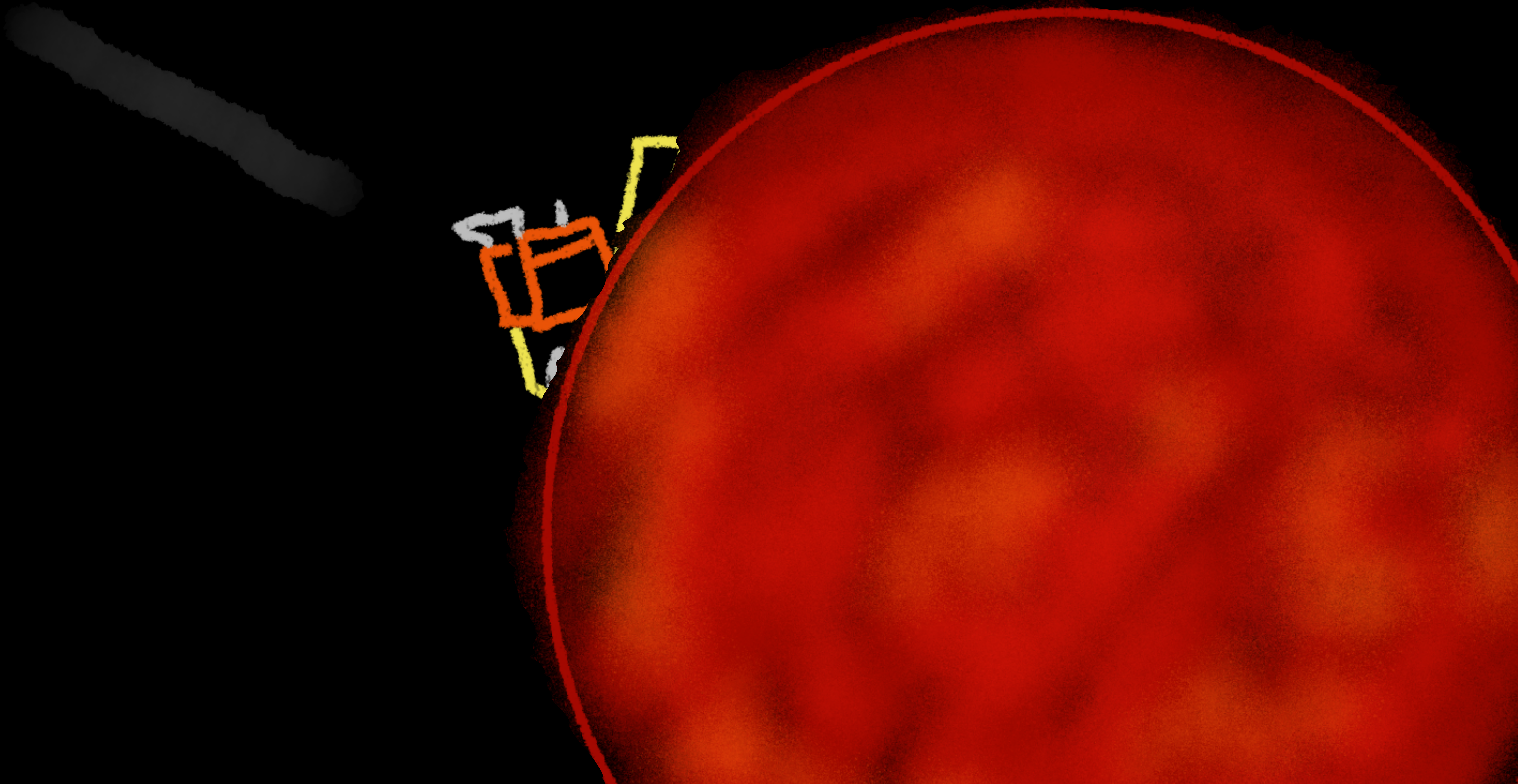


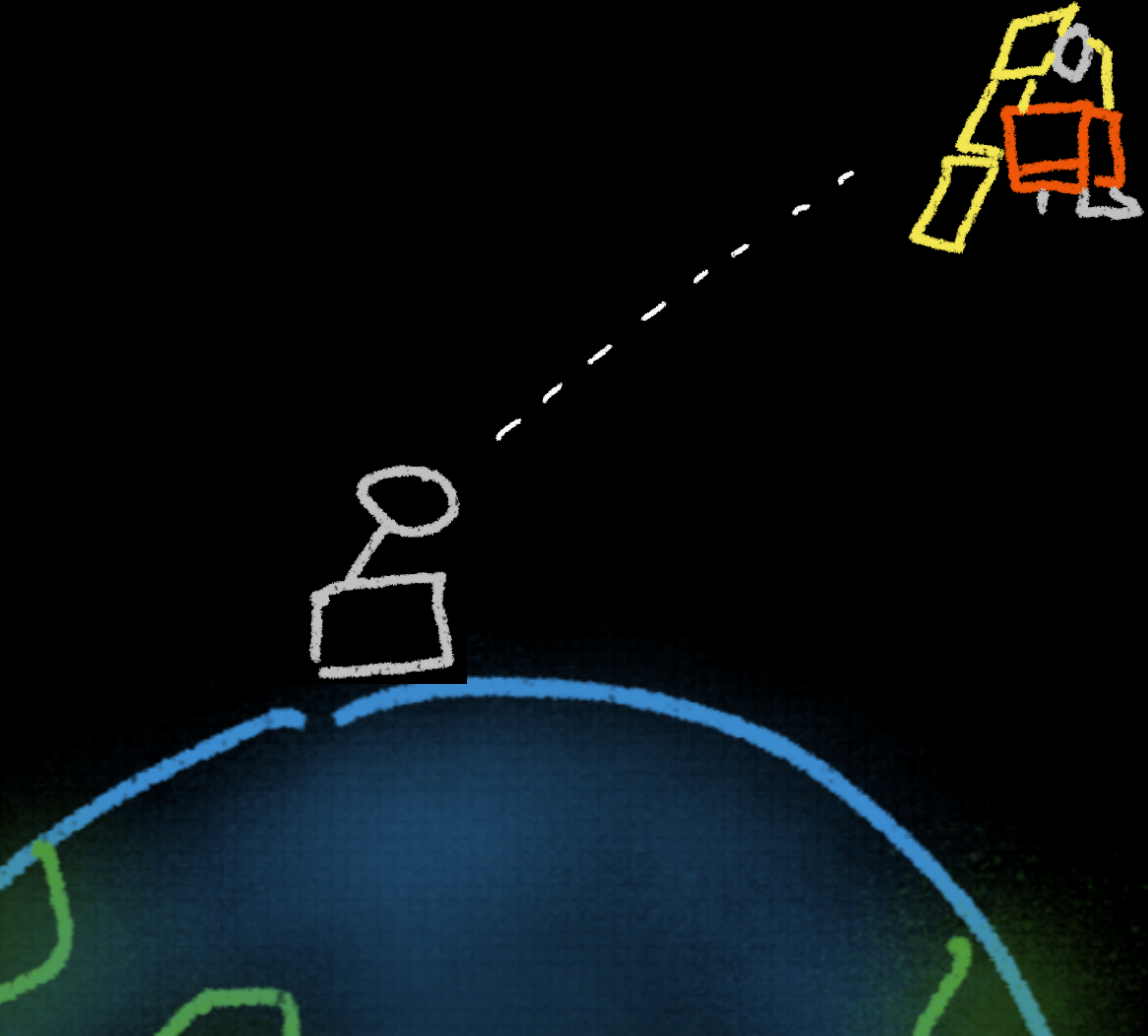


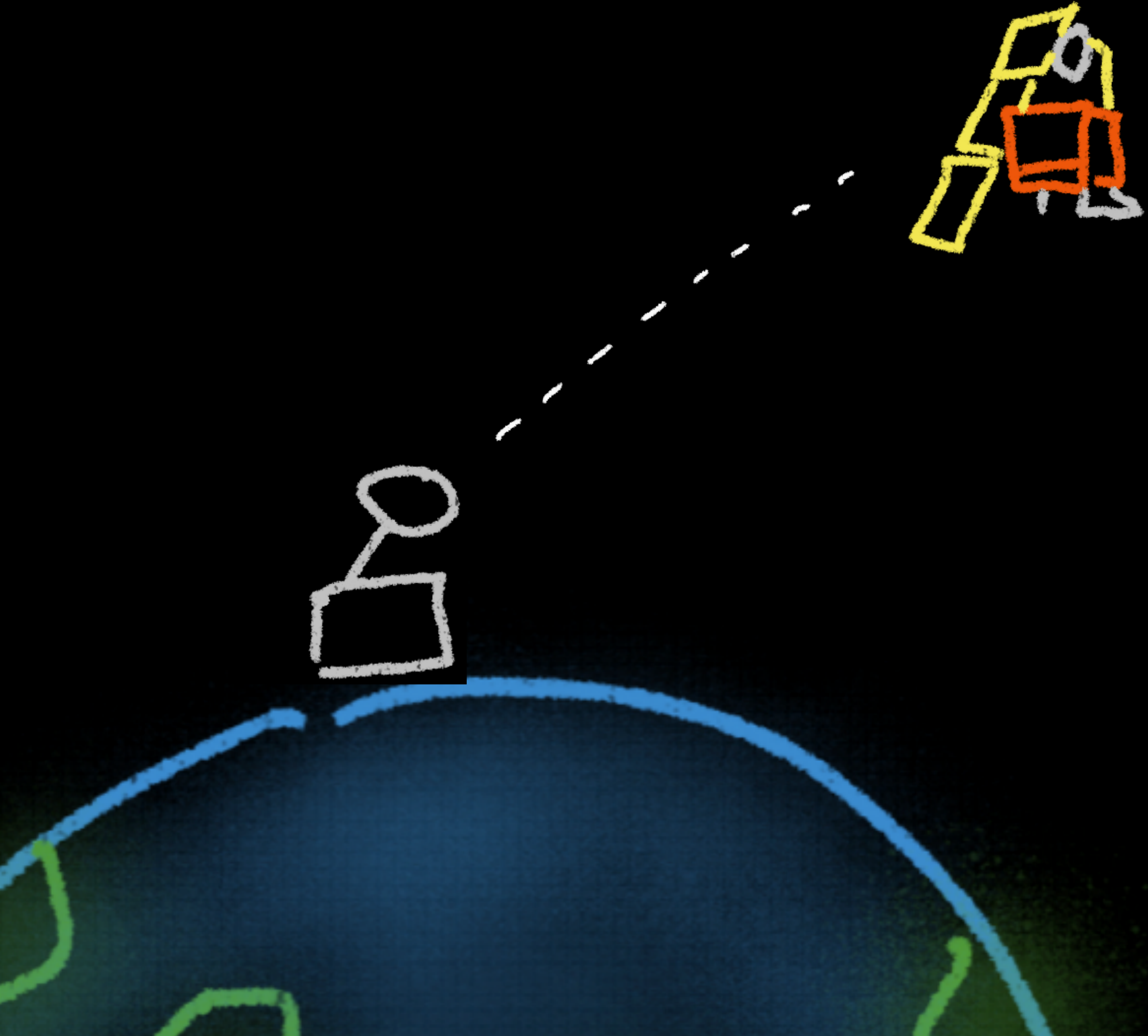
for clarity: this wasn’t a client of mine.

other people’s trenches

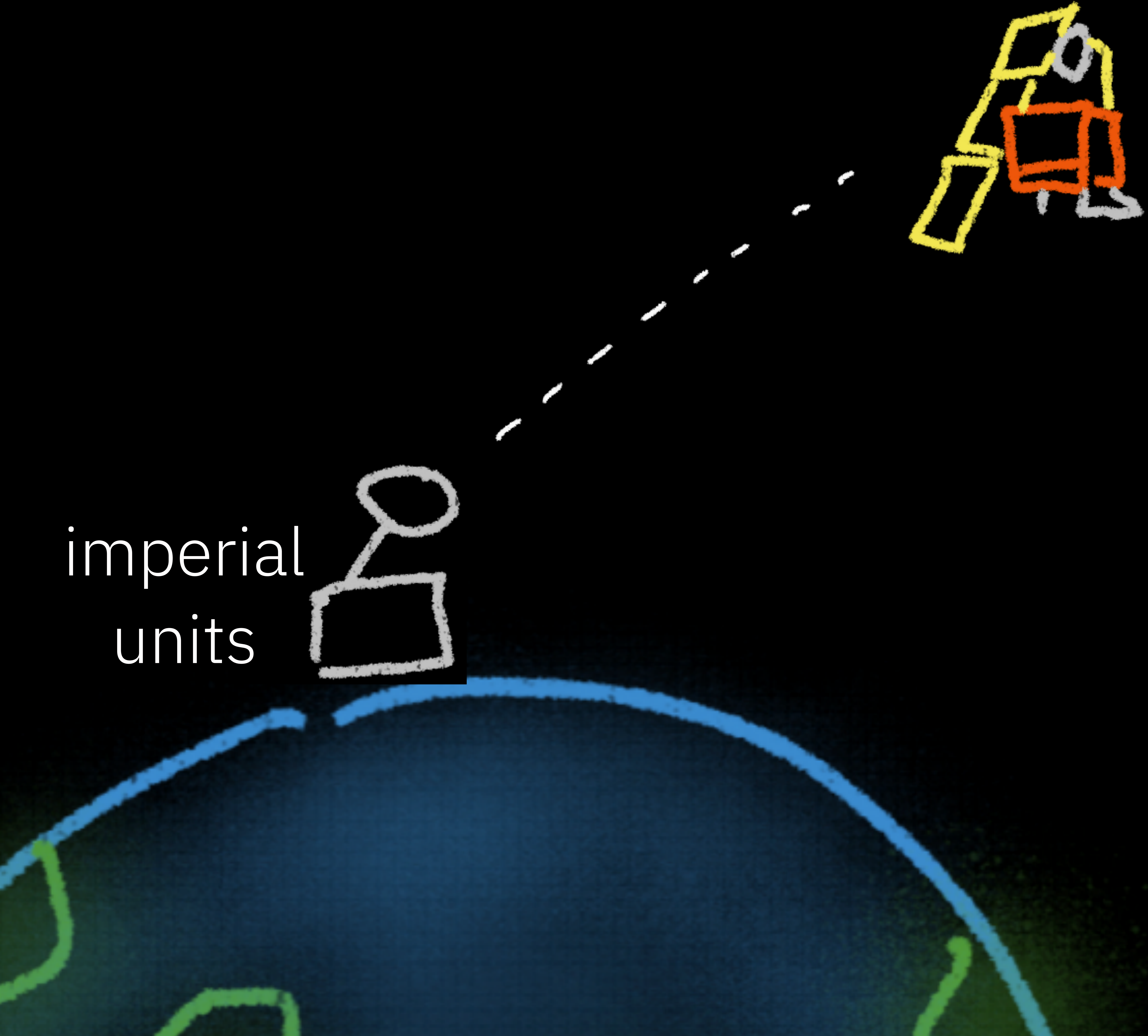
Courtesy NASA/
JPL-Caltech

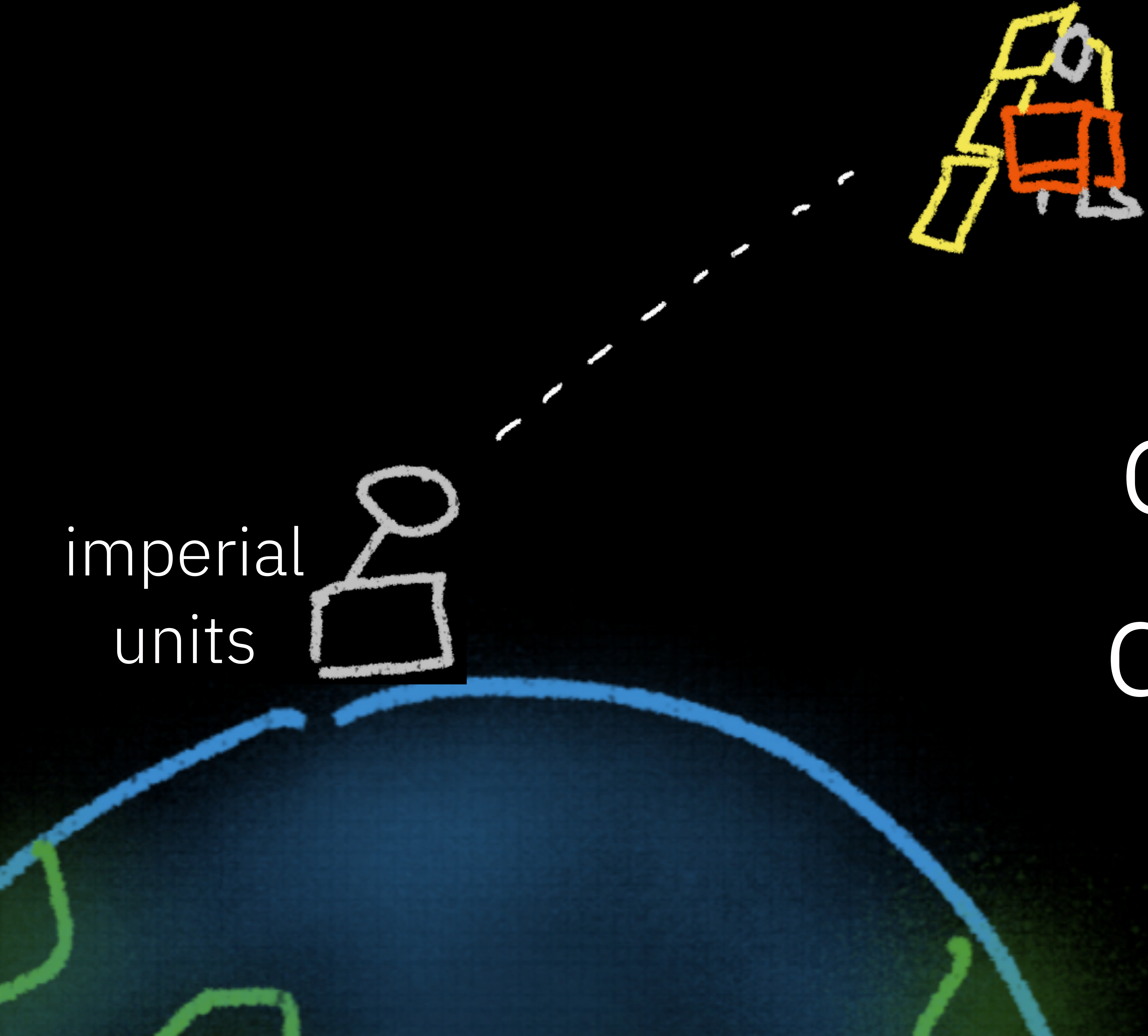






metric units

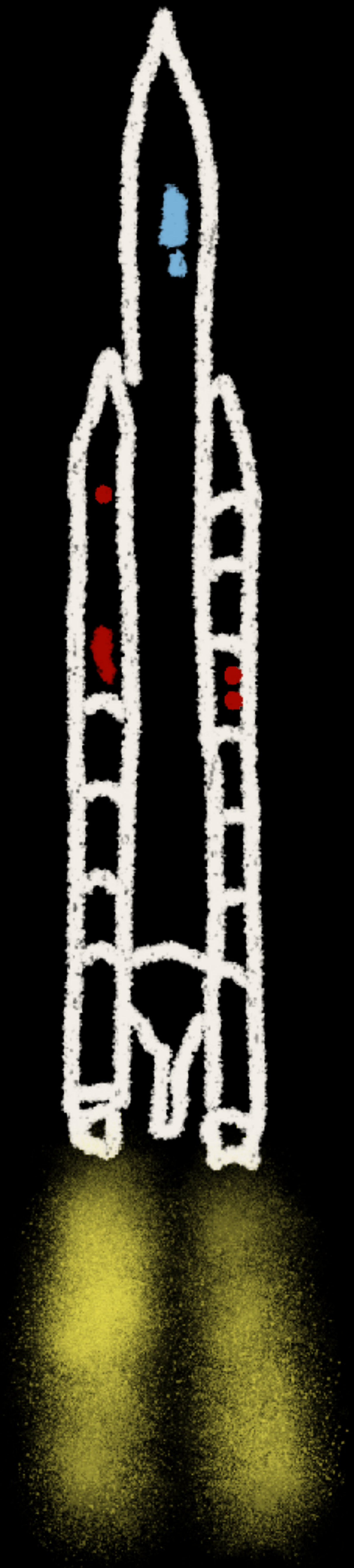

metric units
imperial units



distributing did not help

microservices **need**
consumer-driven contract tests

Cluster + Ariane 5



$370 million loss

https://en.wikipedia.org/wiki/Cluster_(spacecraft)

Cluster + Ariane 5

$370 million loss

https://en.wikipedia.org/wiki/Cluster_(spacecraft)

they tested it …

they tested it ...

but stubbed out one component.

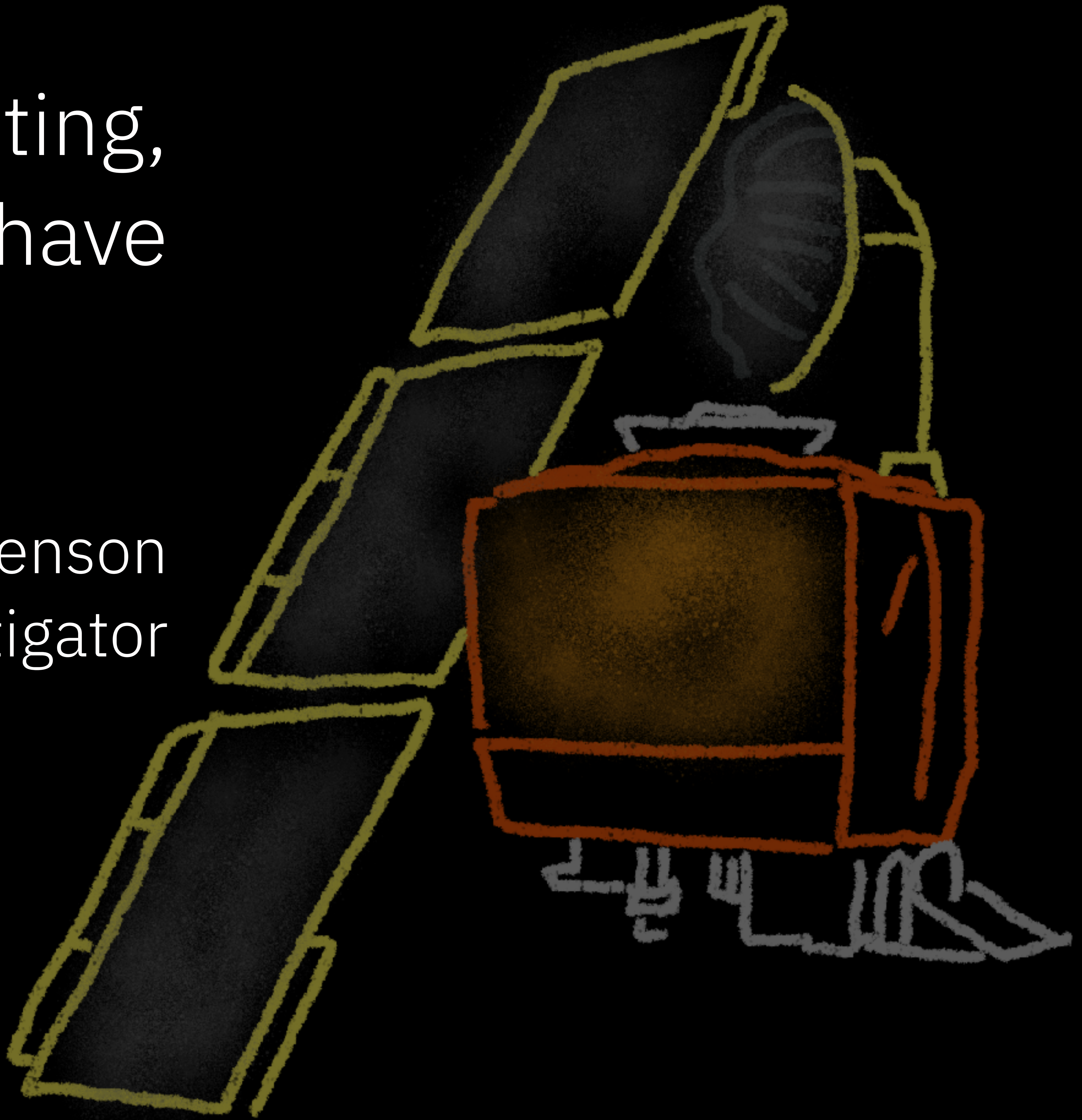
they tested it …

but stubbed out one component.

that component was the one that broke.

"Had we done end-to-end testing, we believe this error would have been caught."

Arthur Stephenson
Chief Investigator



microservices **need**

automated integration tests

“we have a CI/CD”

CI/CD is something you **do**, it's 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 …. ”

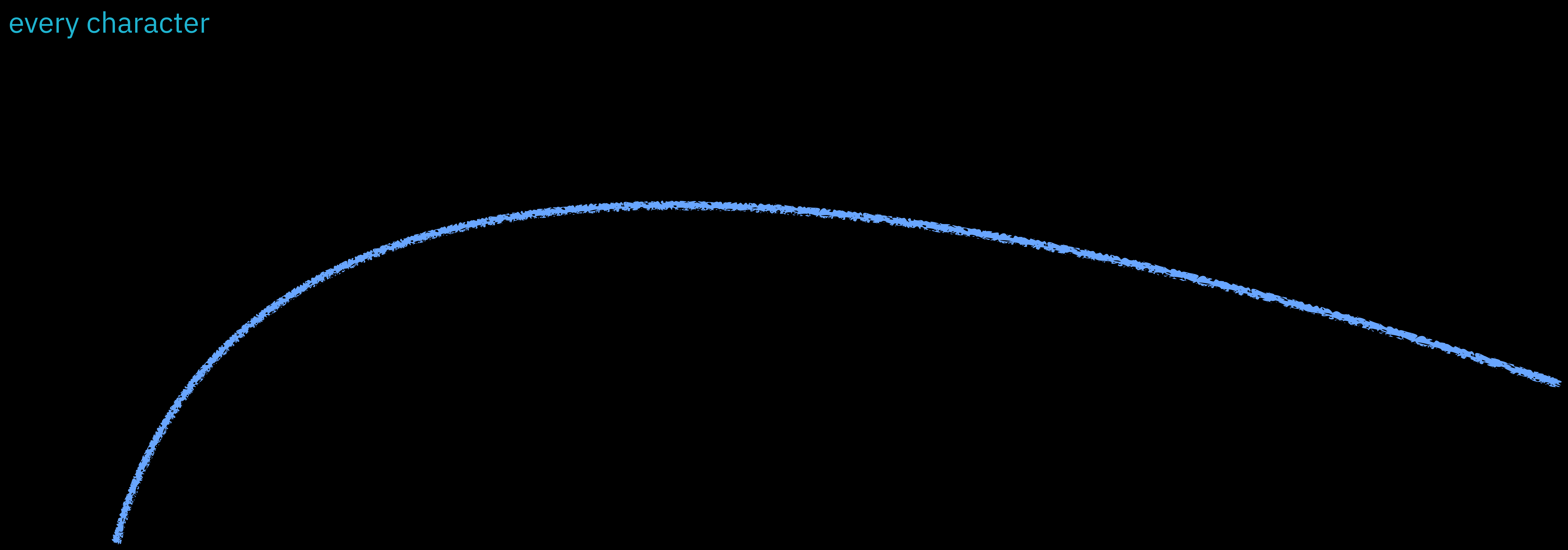
continuous.

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

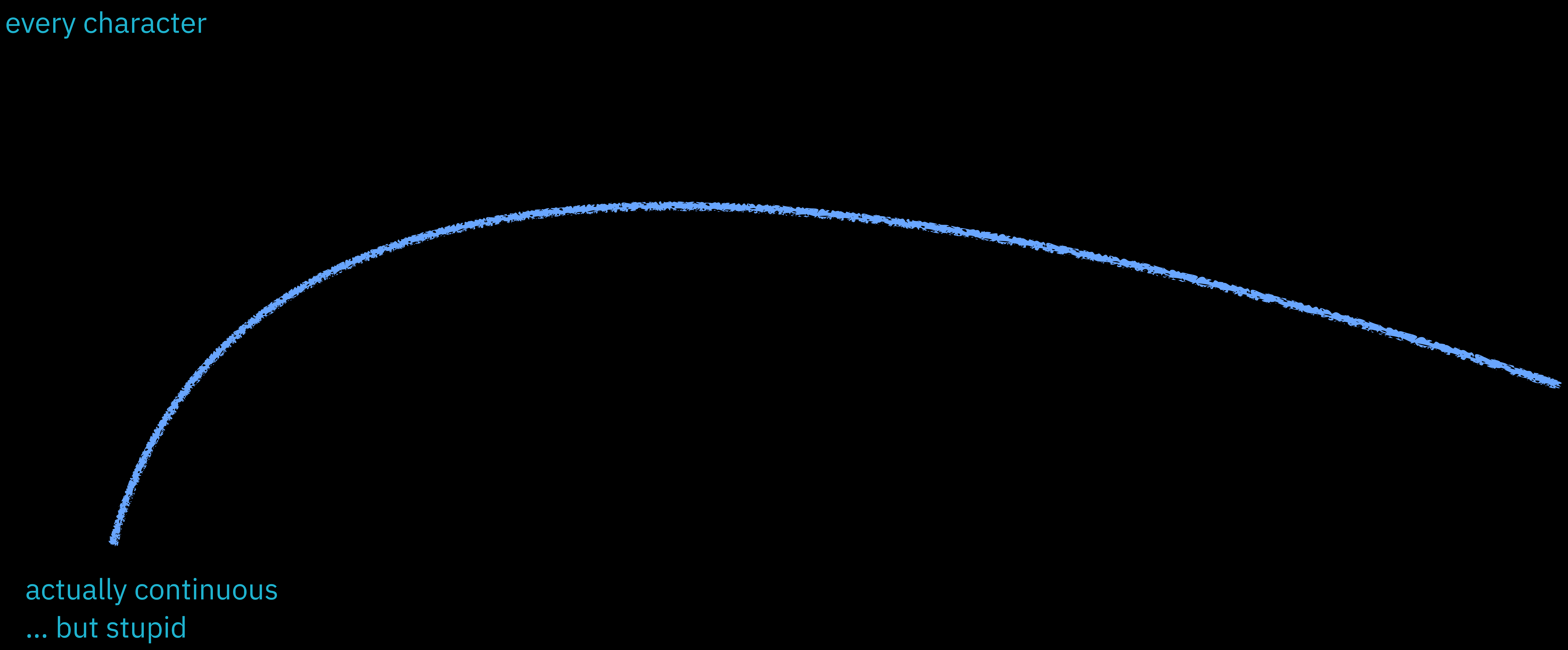
how often should you push to master?

how often should you integrate?

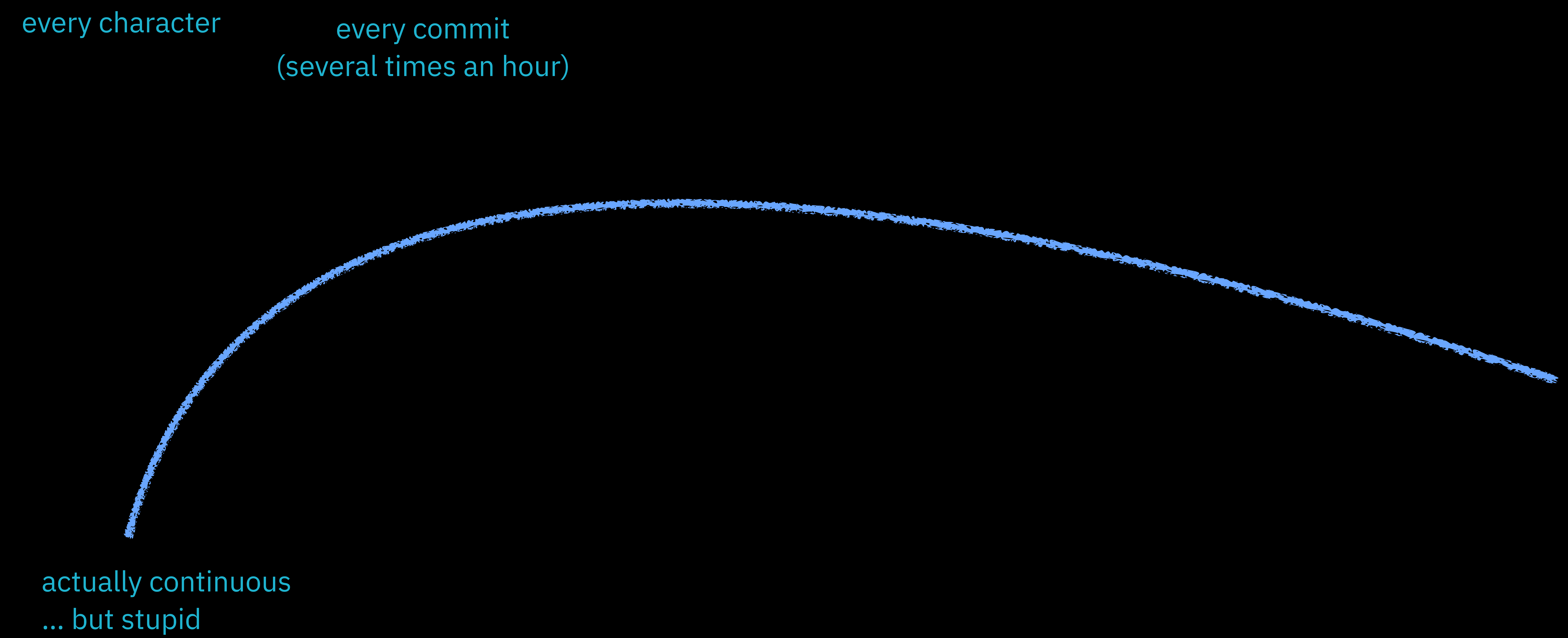
how often should you integrate?



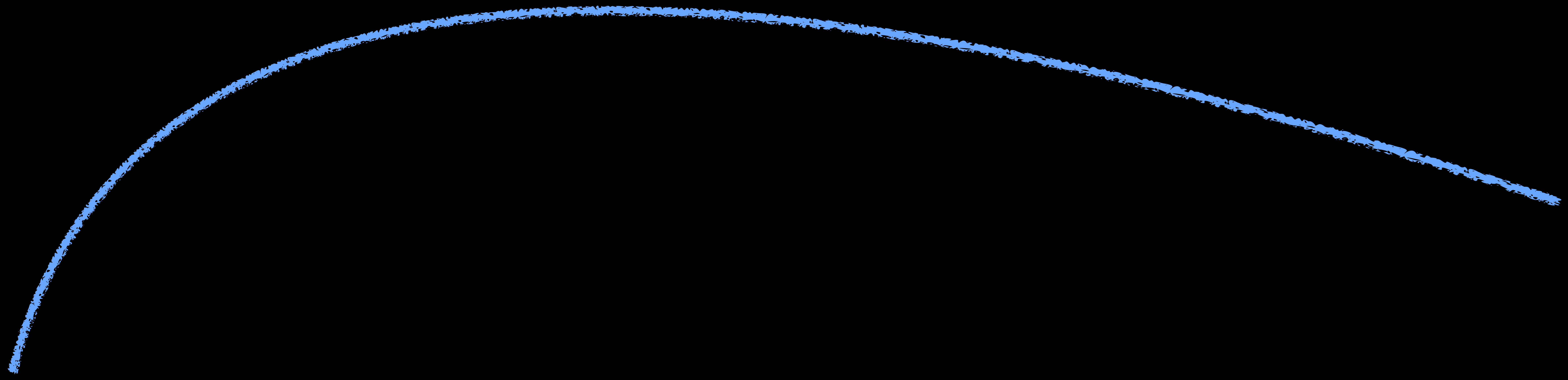
how often should you integrate?



how often should you integrate?



how often should you integrate?



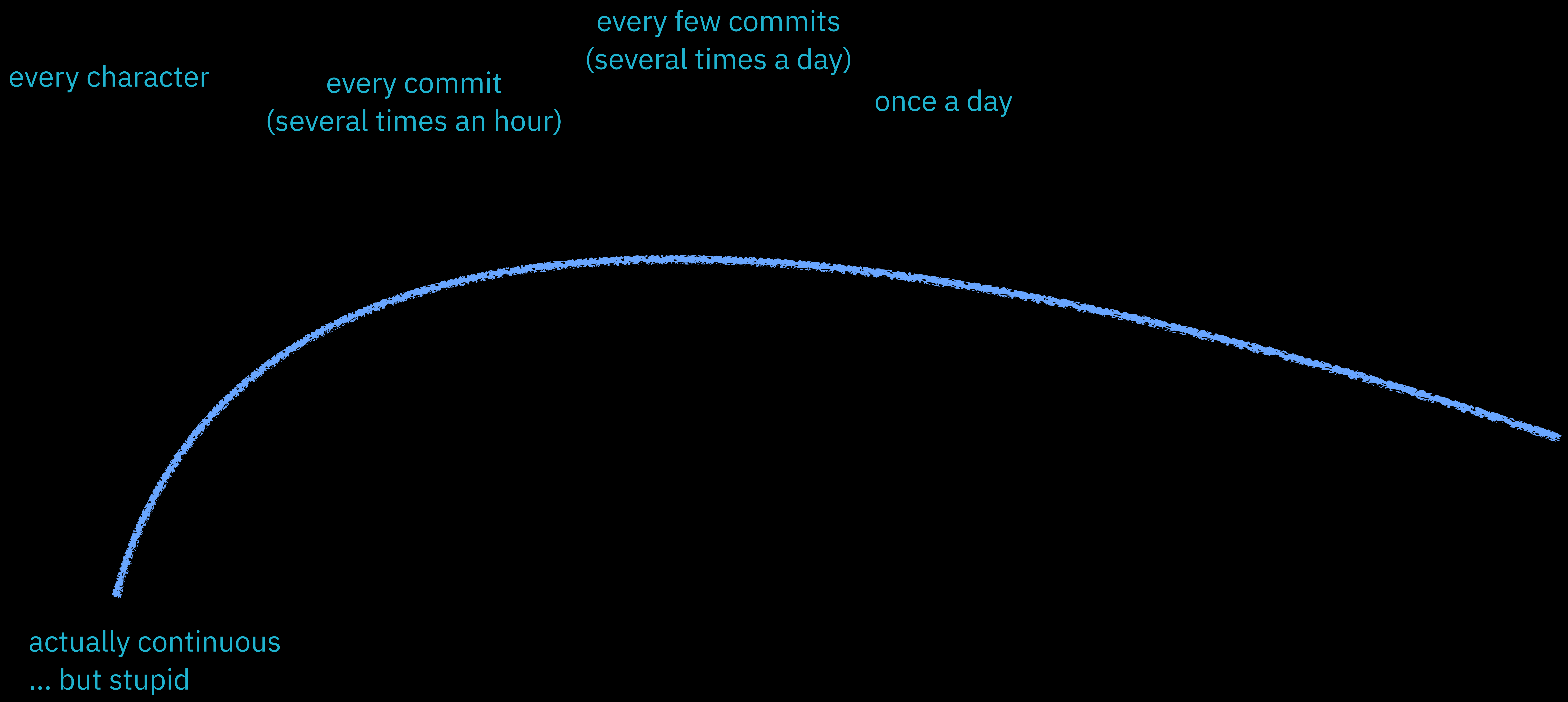
every few commits
(several times a day)

every character

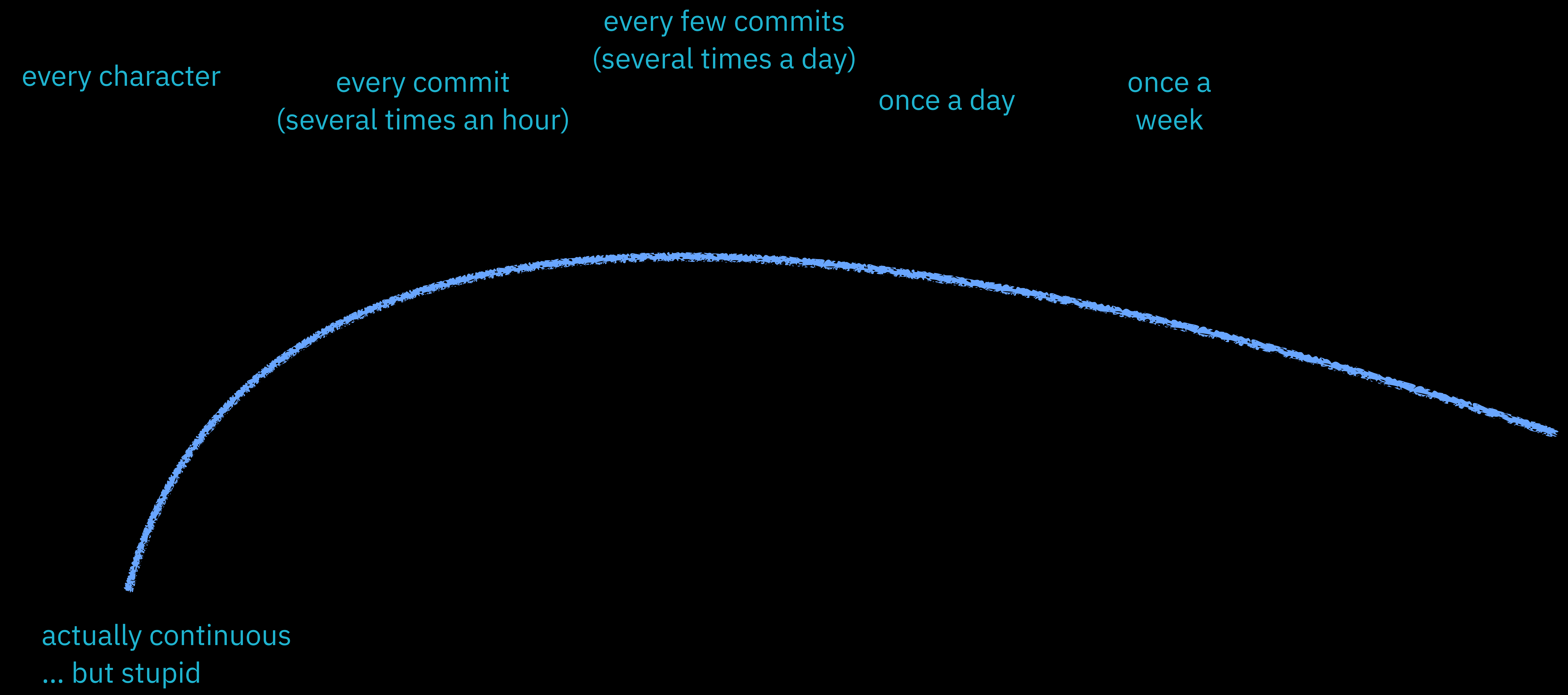
every commit
(several times an hour)

actually continuous
... but stupid

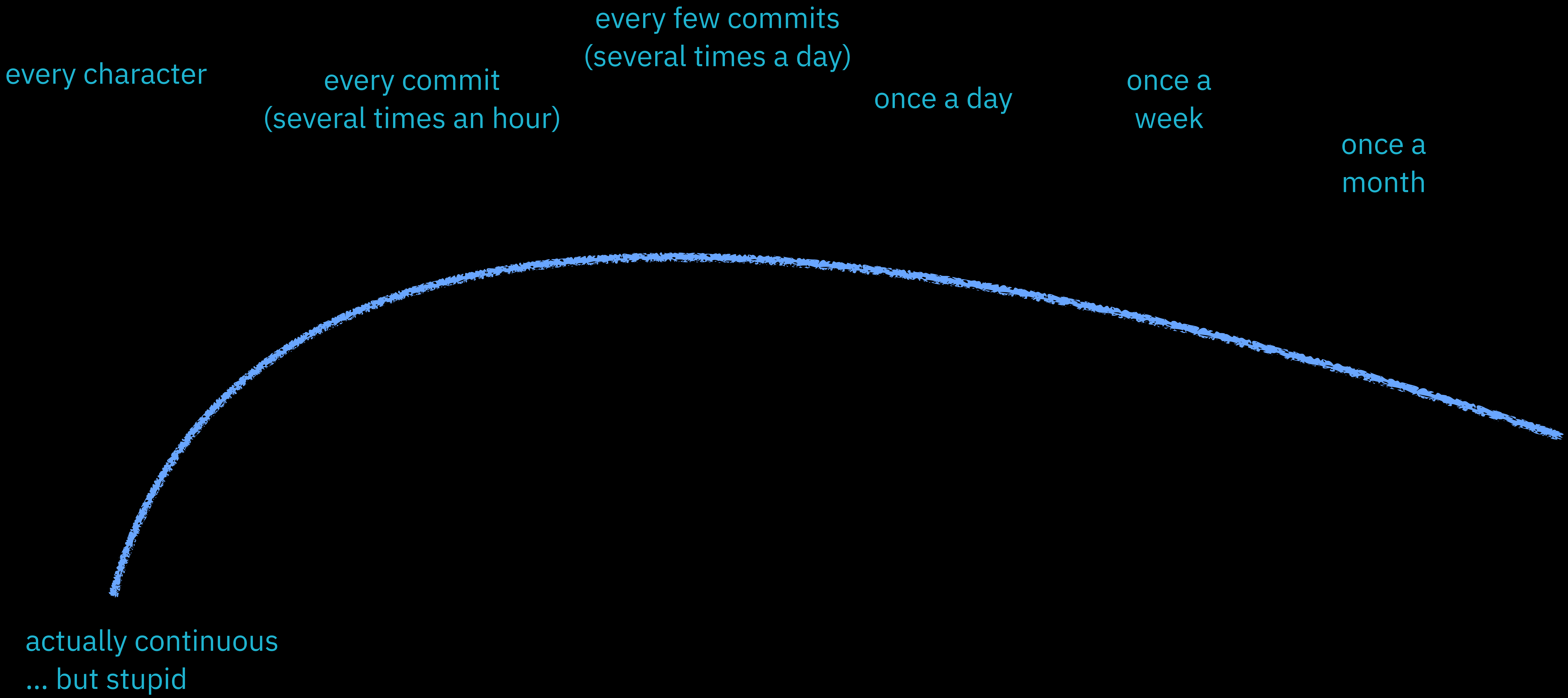
how often should you integrate?



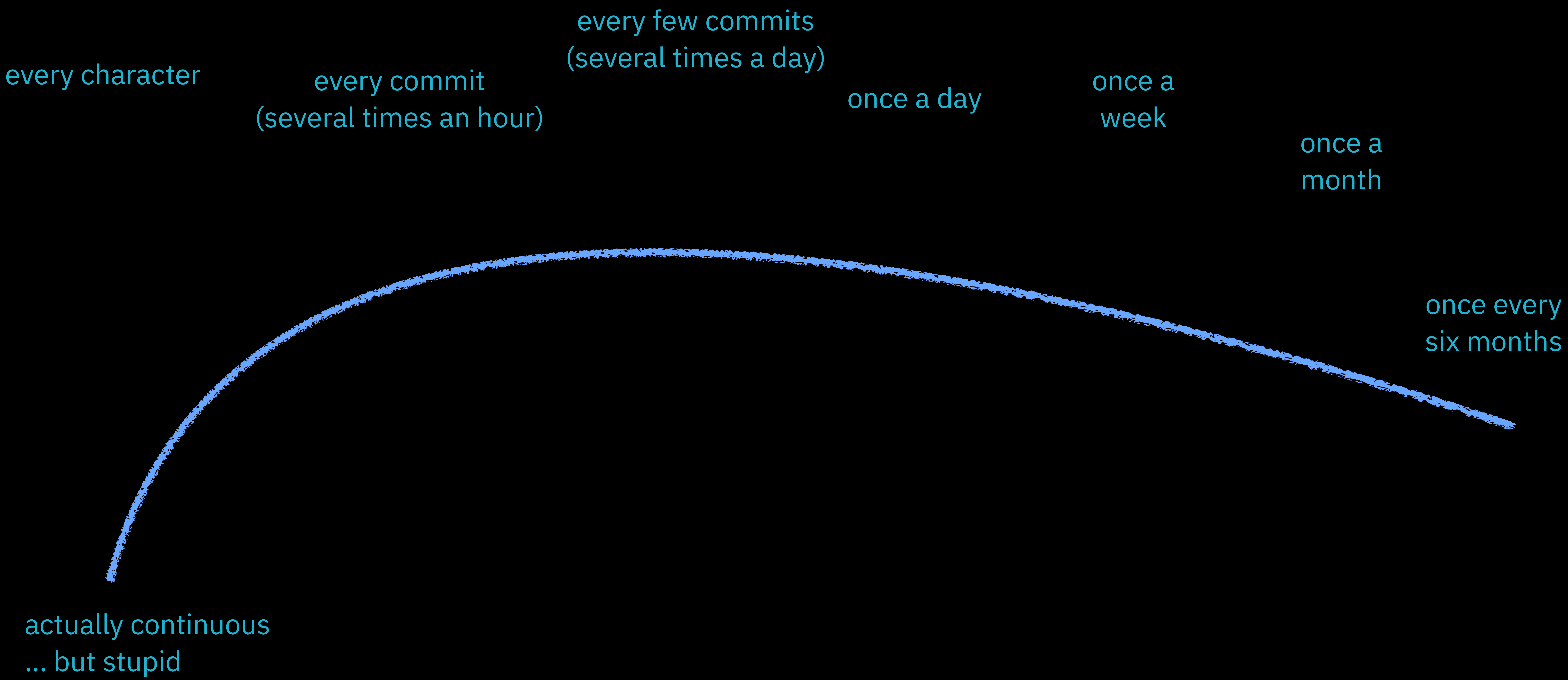
how often should you integrate?



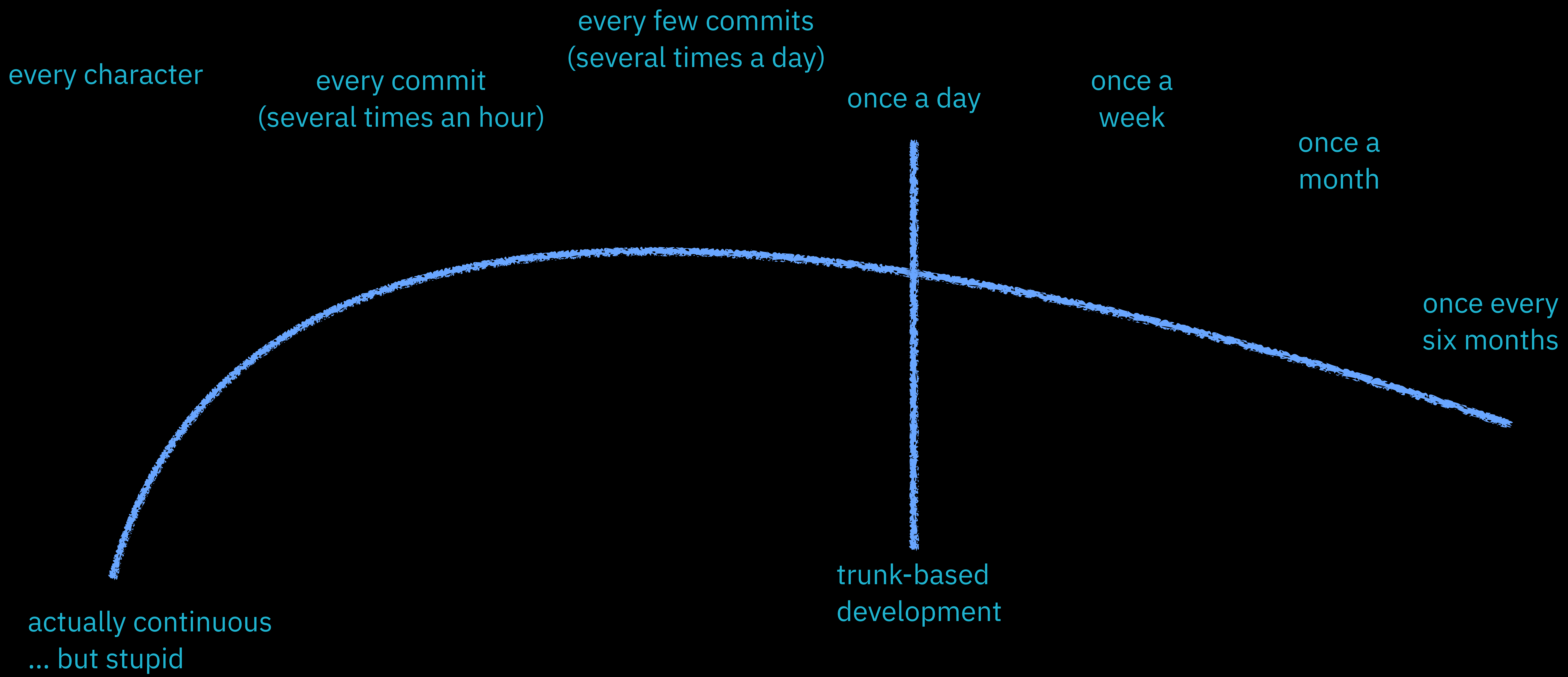
how often should you integrate?



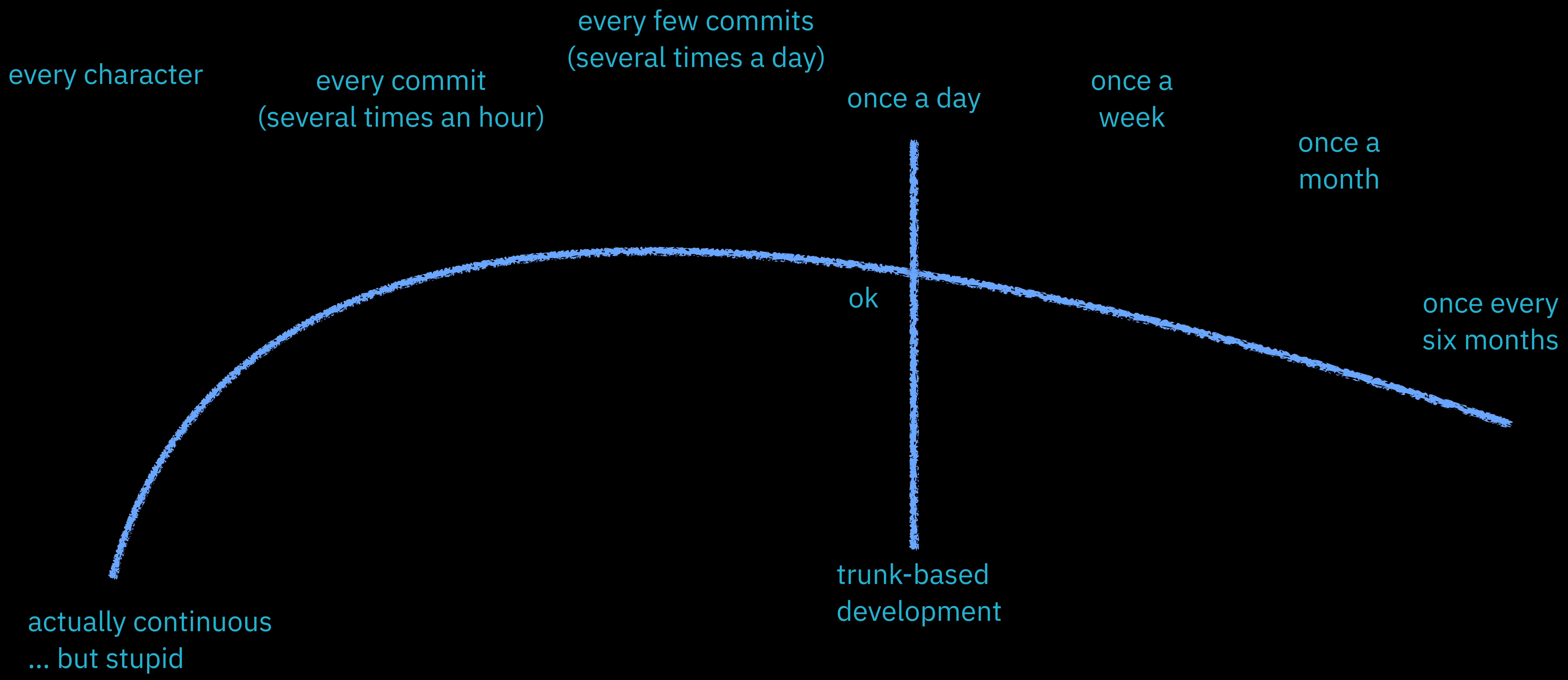
how often should you integrate?



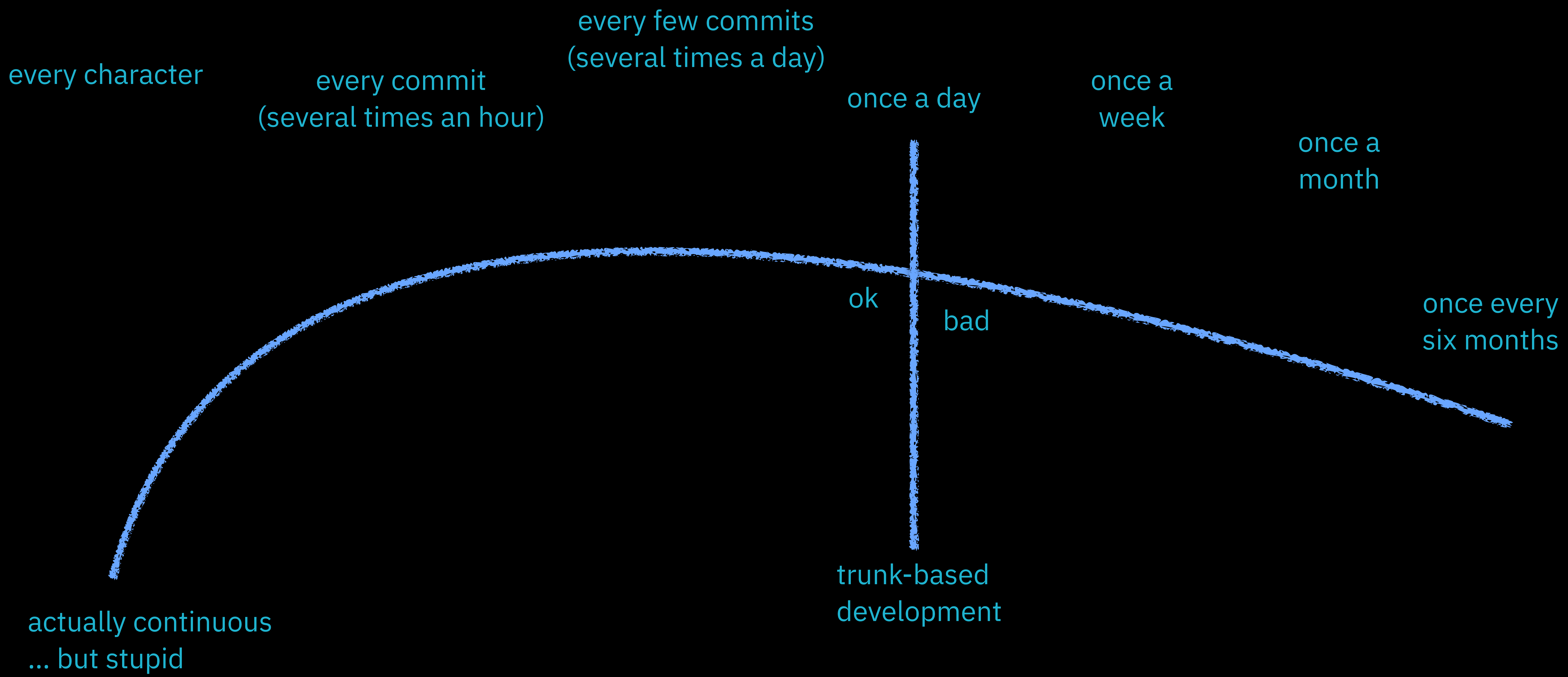
how often should you integrate?



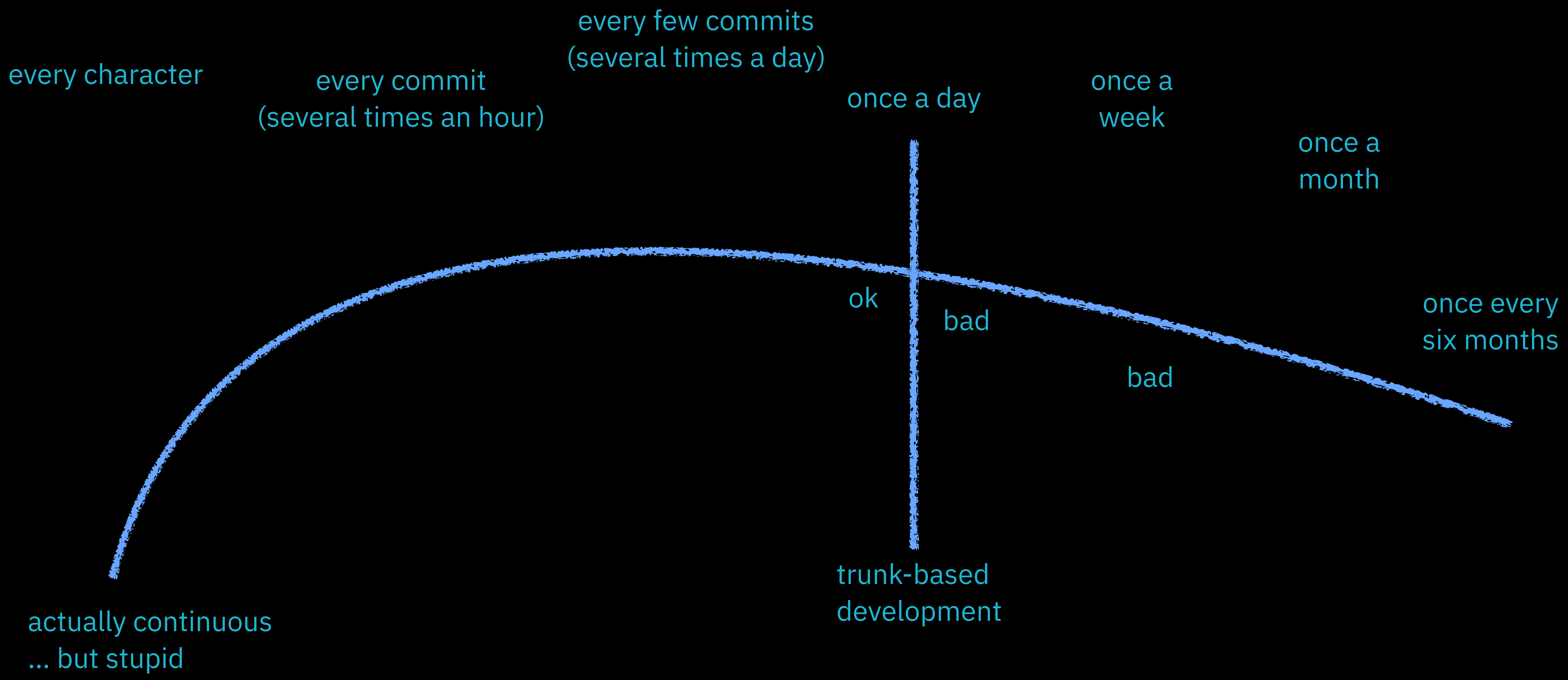
how often should you integrate?



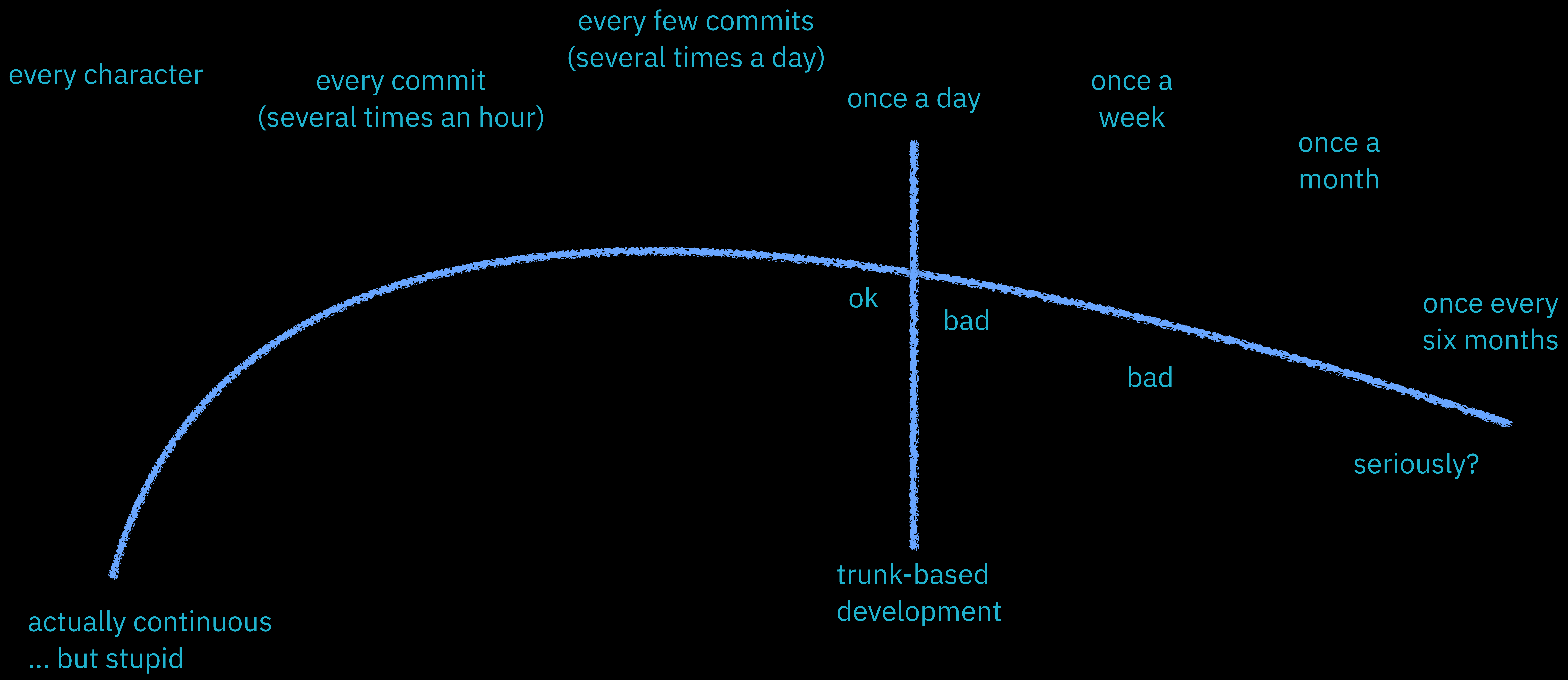
how often should you integrate?



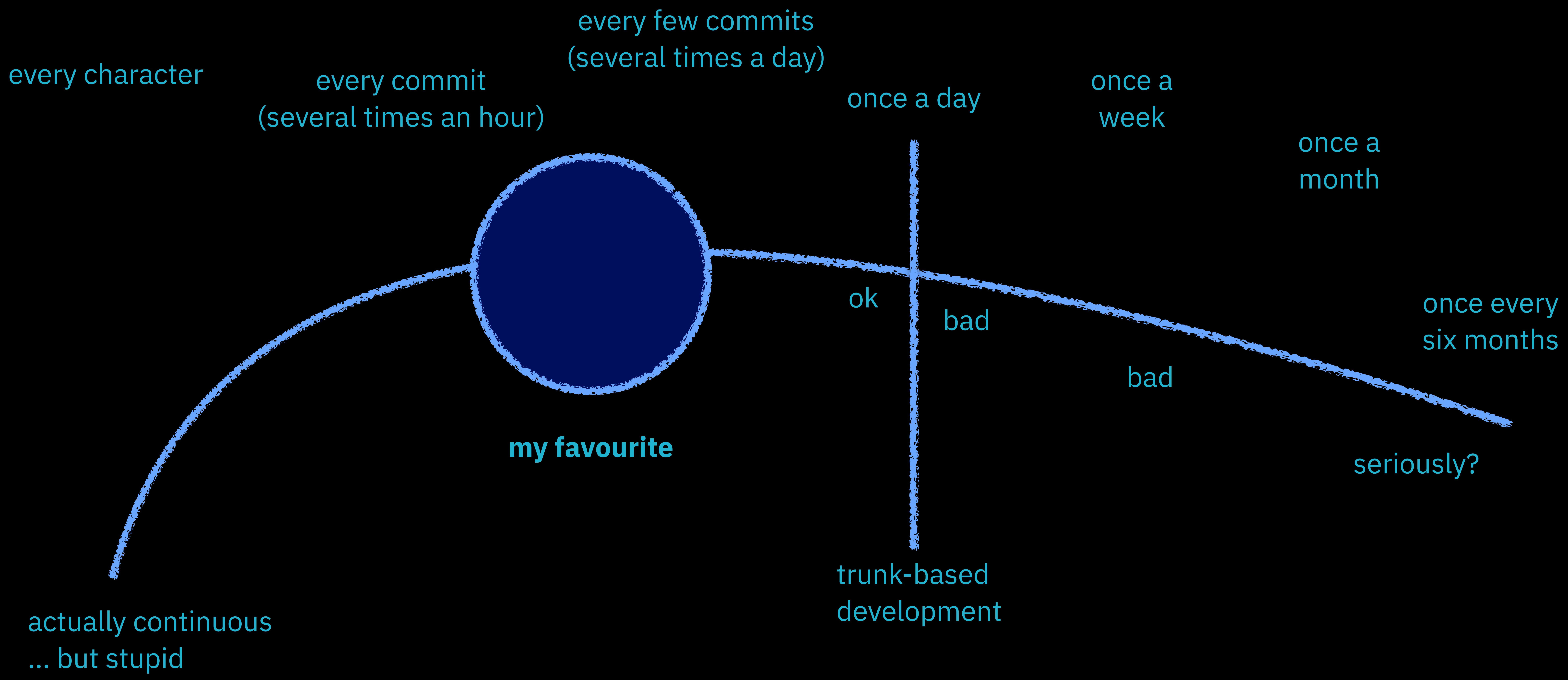
how often should you integrate?



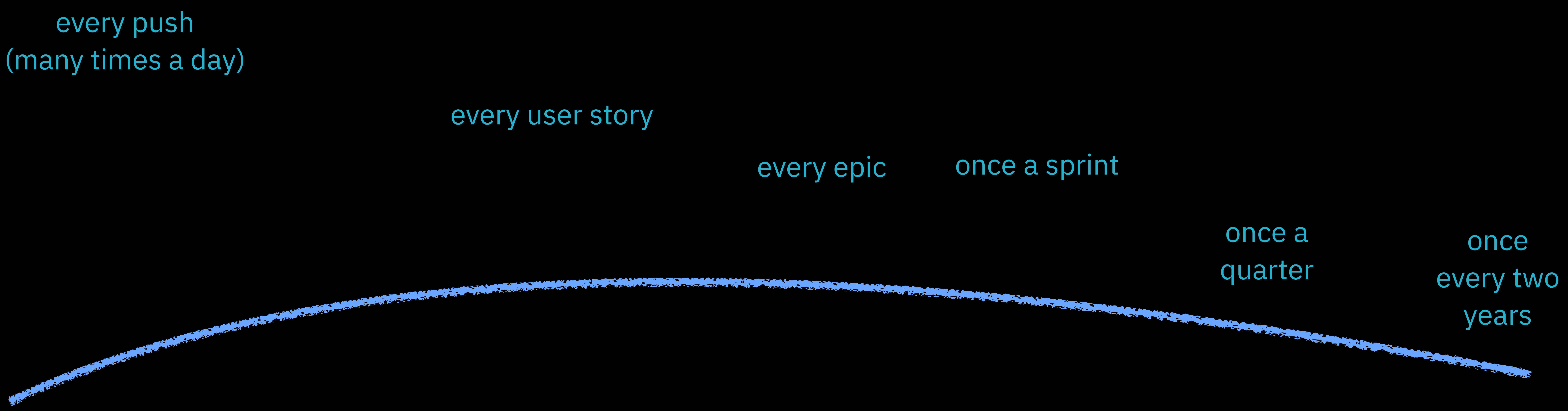
how often should you integrate?



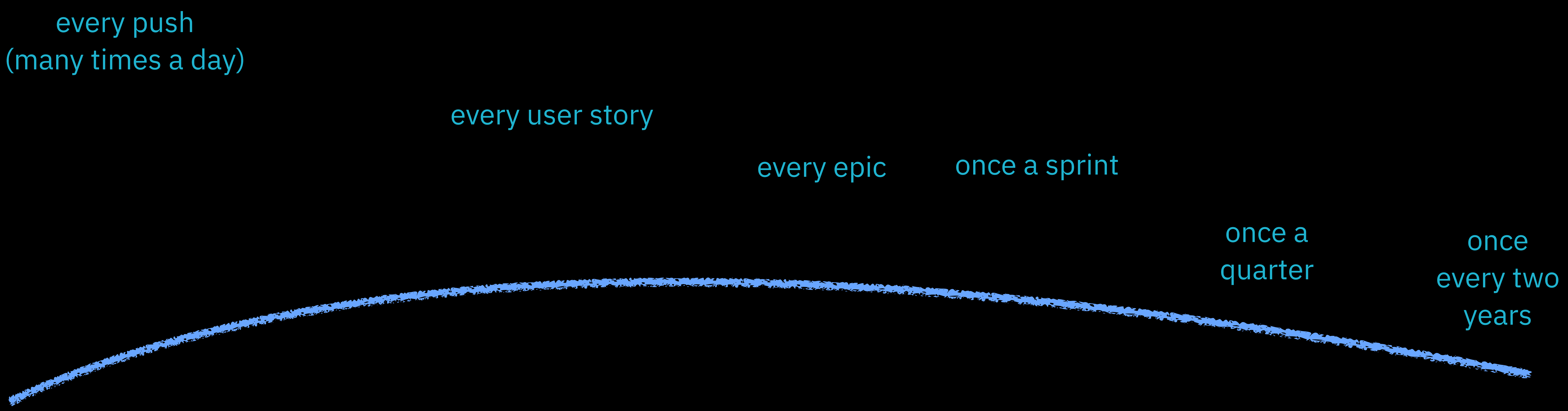
how often should you integrate?



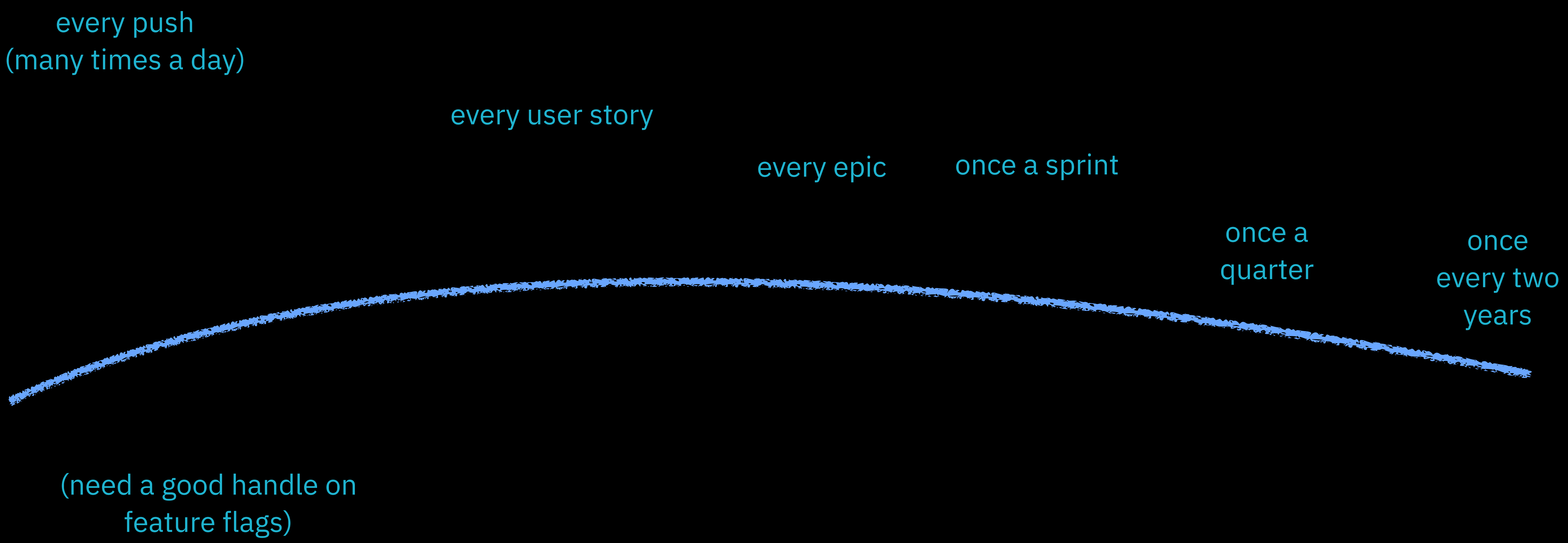
how often should you release?



how often should you deploy?

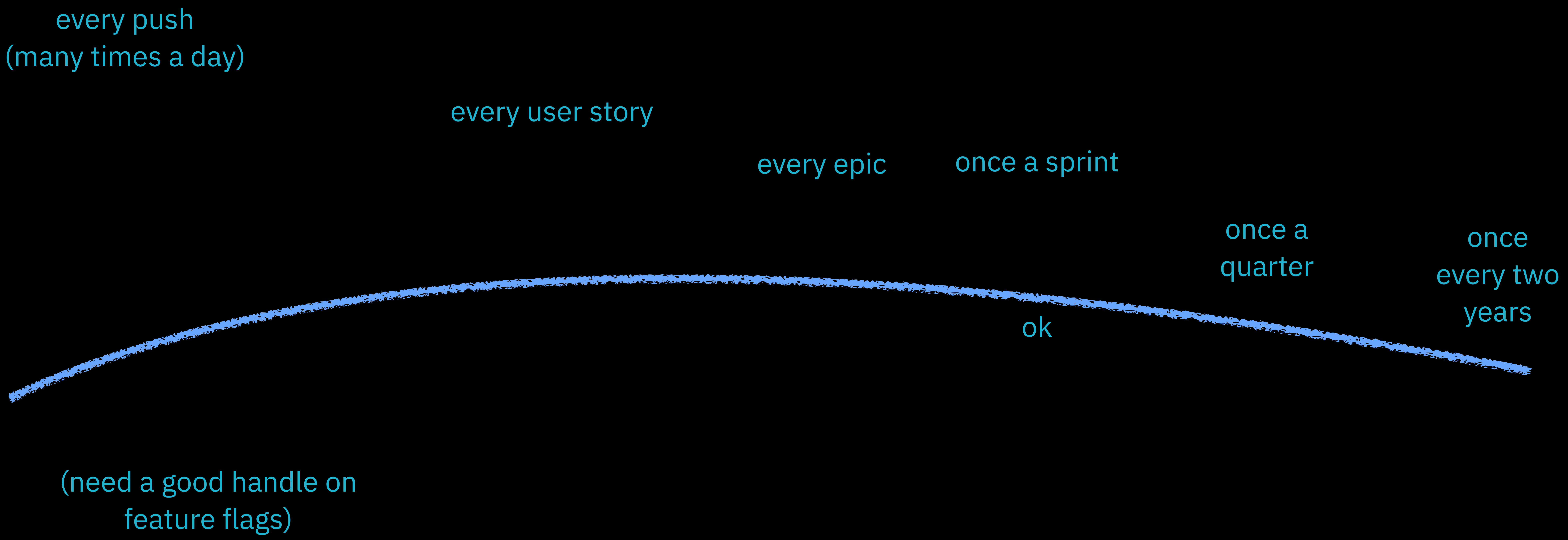


how often should you deploy?

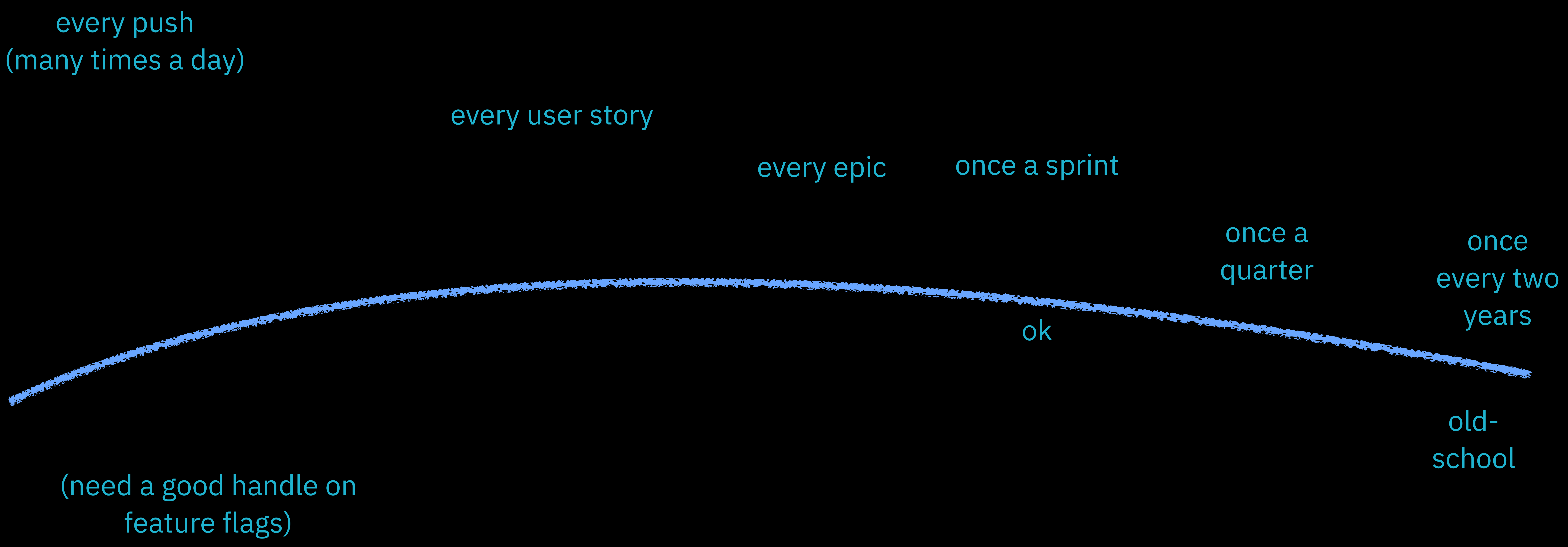


#IBMGarage @holly_cummins

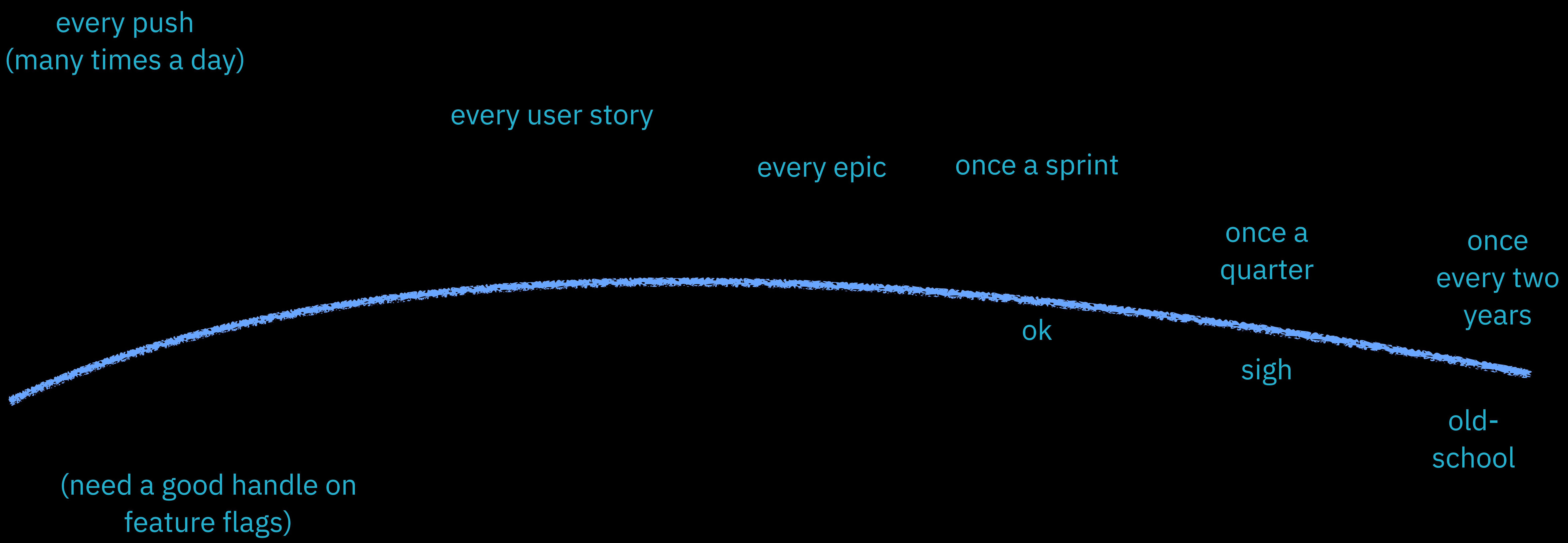
how often should you deploy?



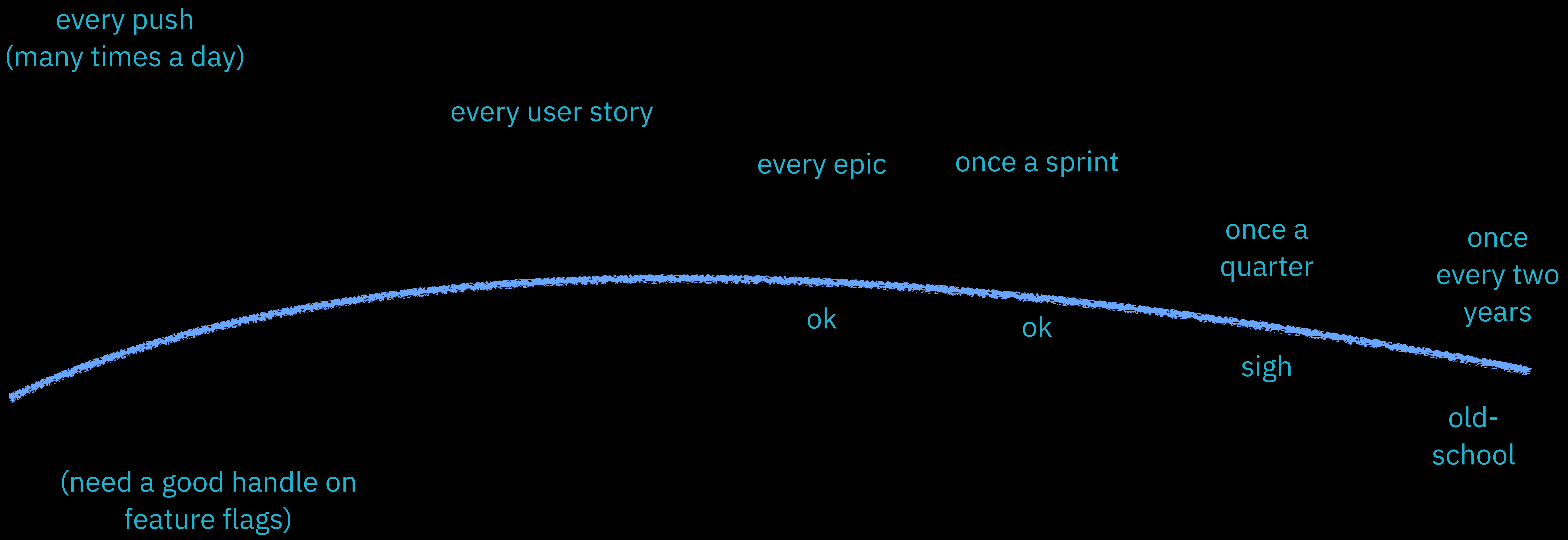
how often should you deploy?



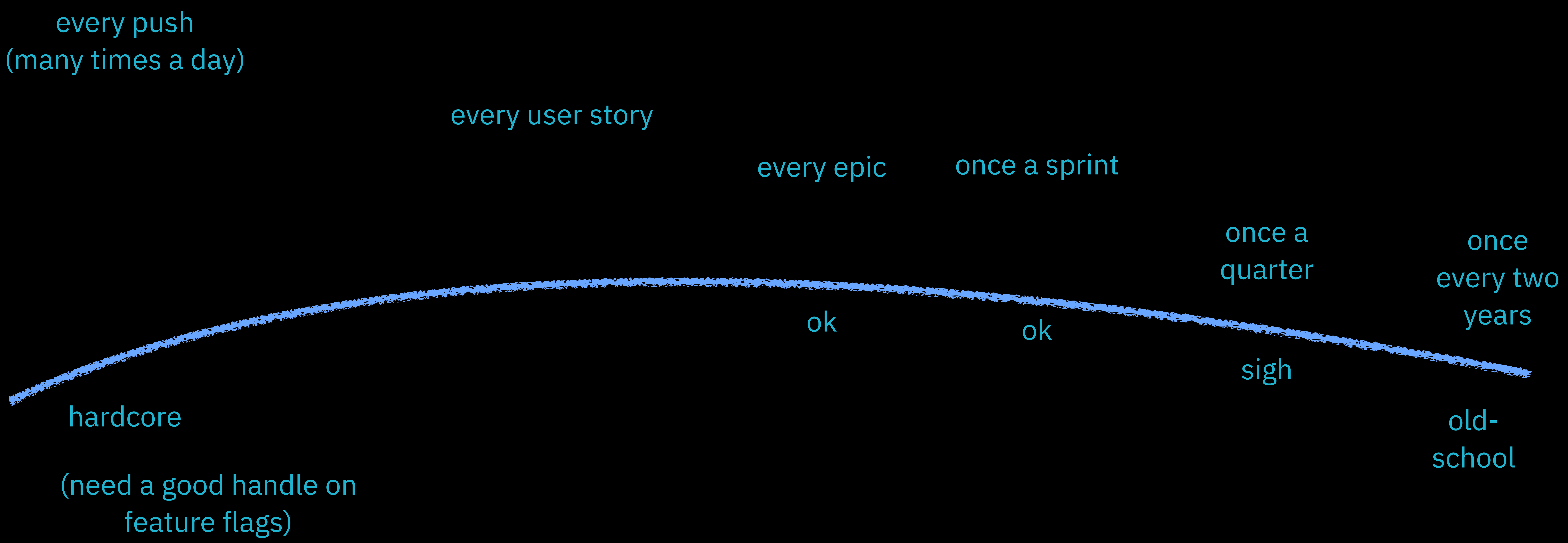
how often should you deploy?



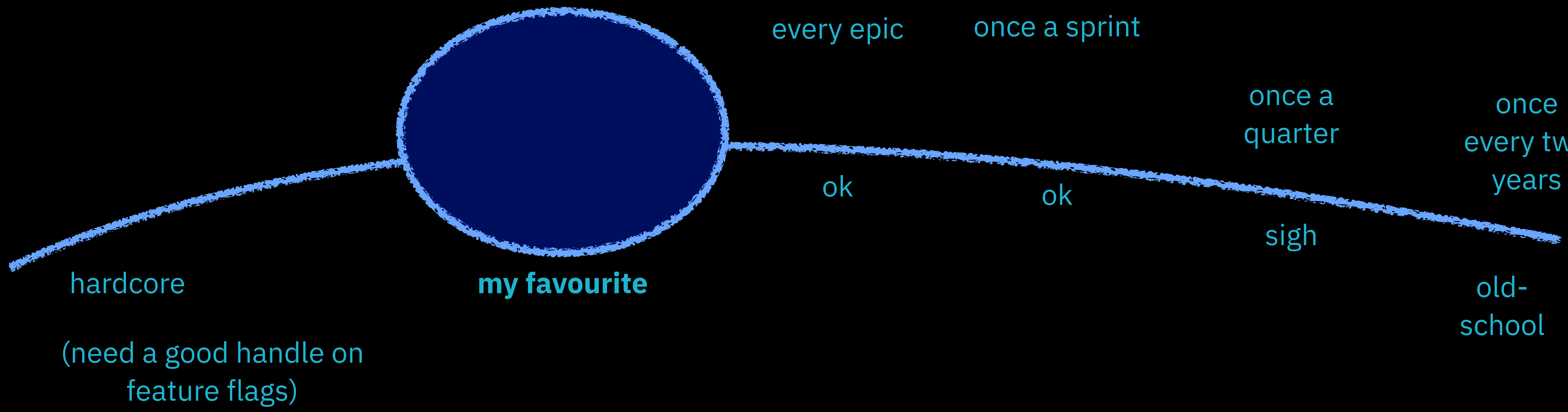
how often should you deploy?



how often should you deploy?



how often should you deploy?

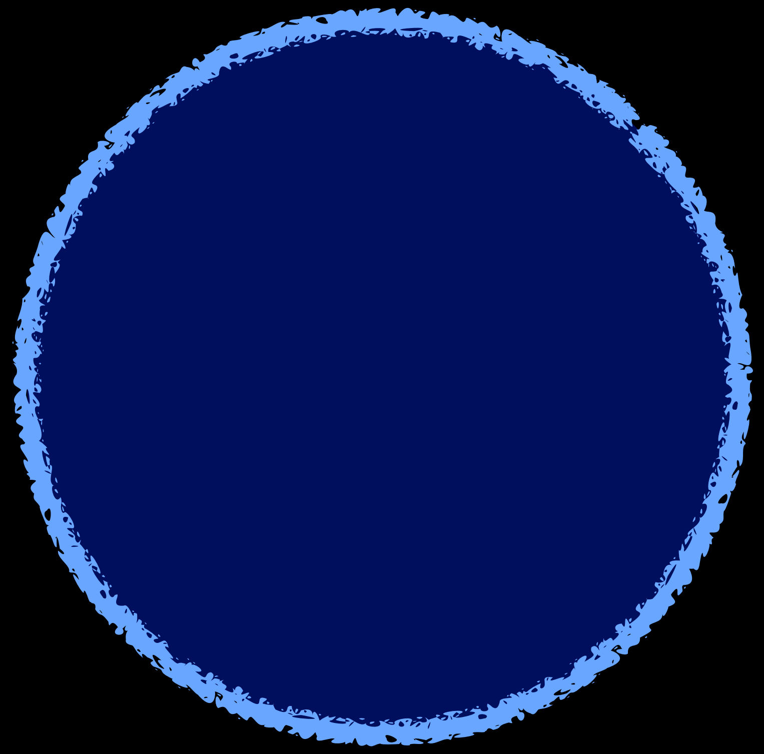


how often should you test in staging?

how often should you deliver?

how often should you deliver?

every push



my favourite

“we can’t actually **release** this.”

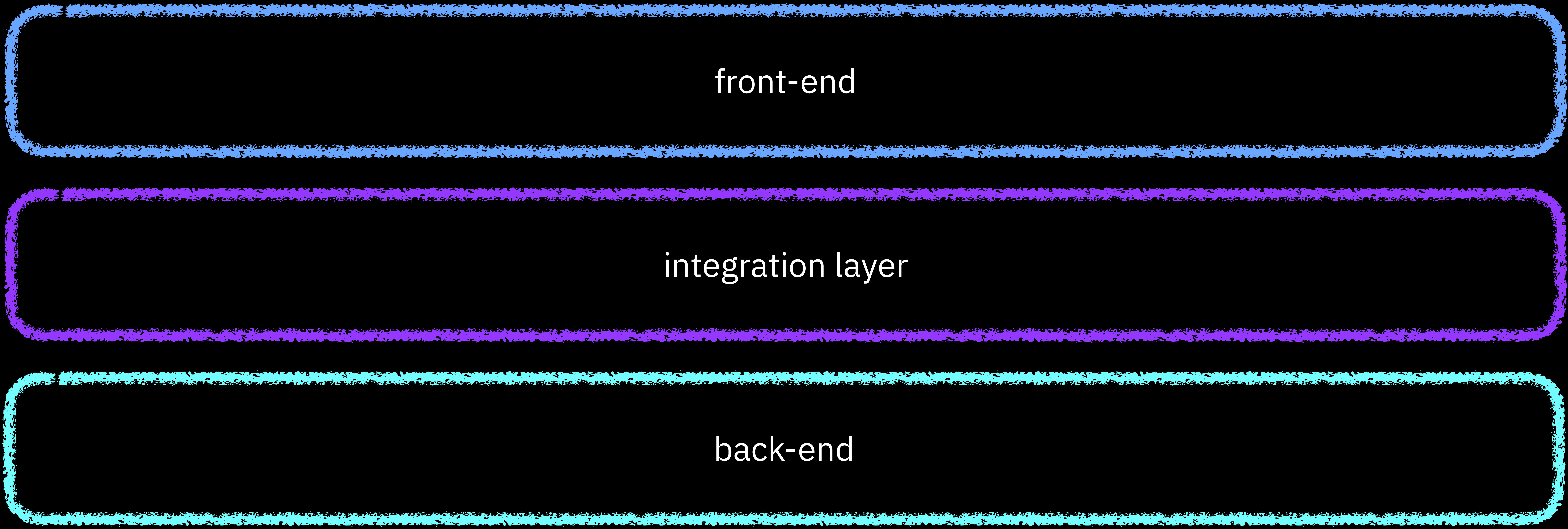
why?

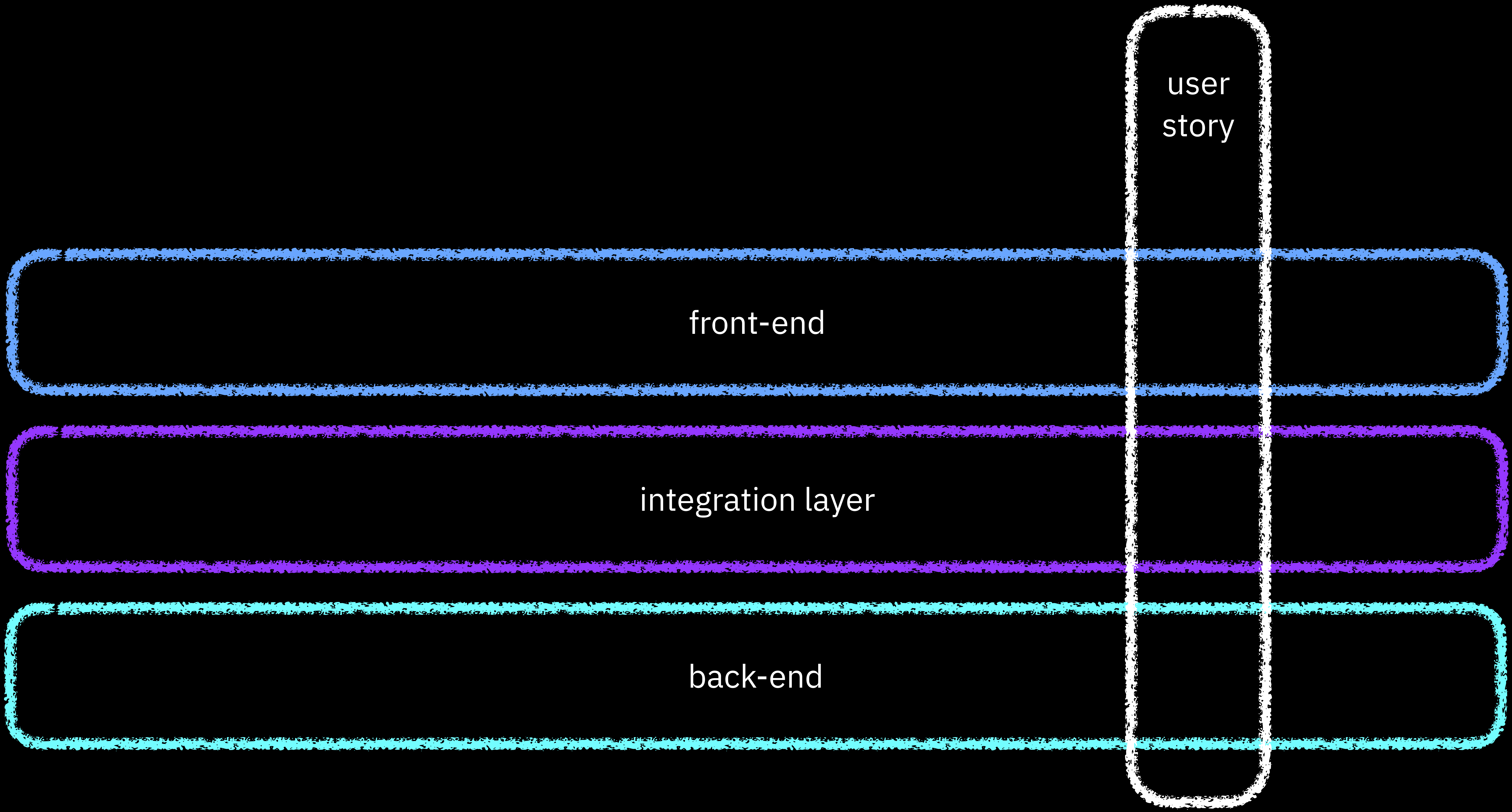
what’s stopping more frequent deploys?

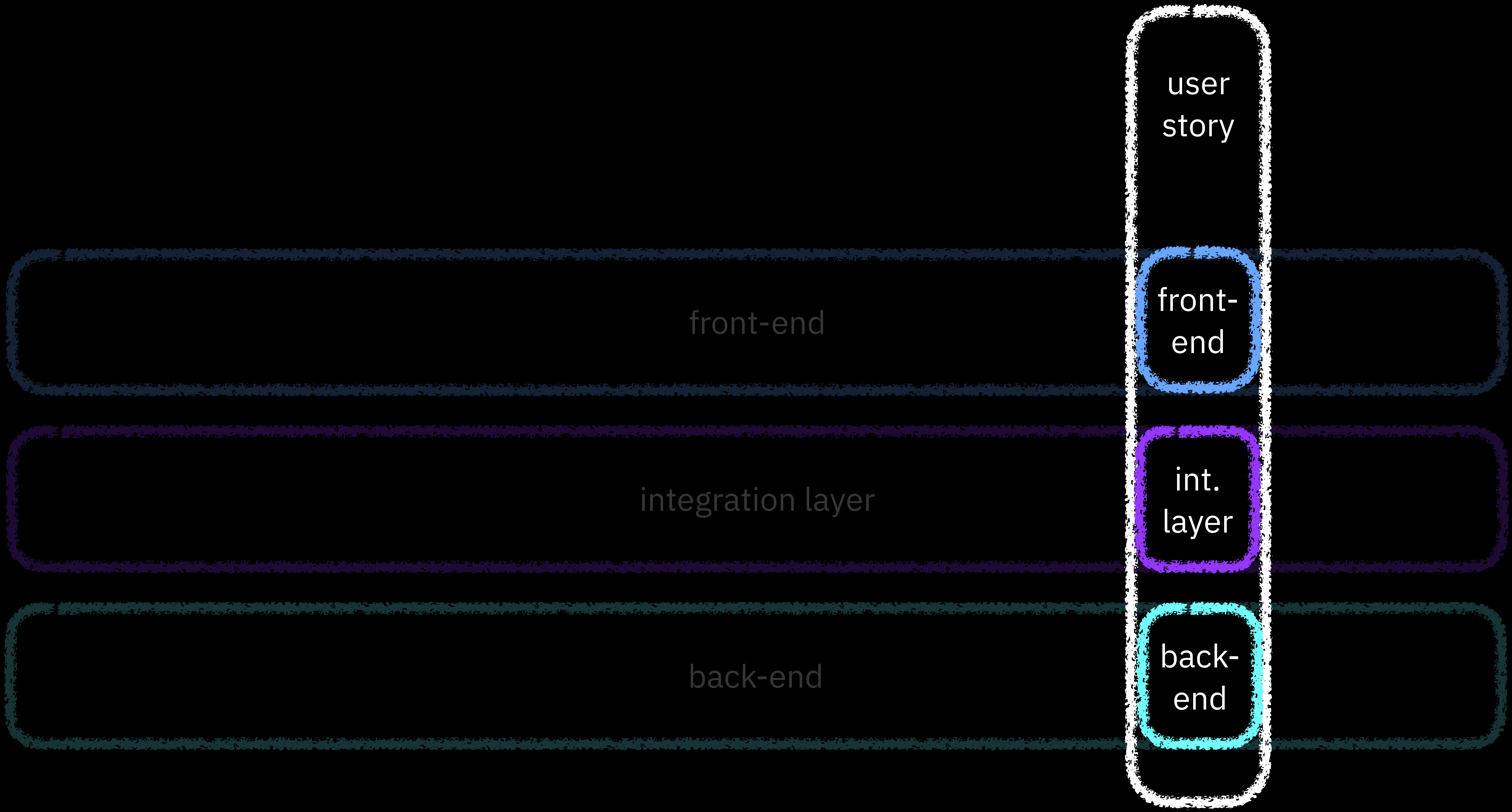
“we can’t release this microservice…

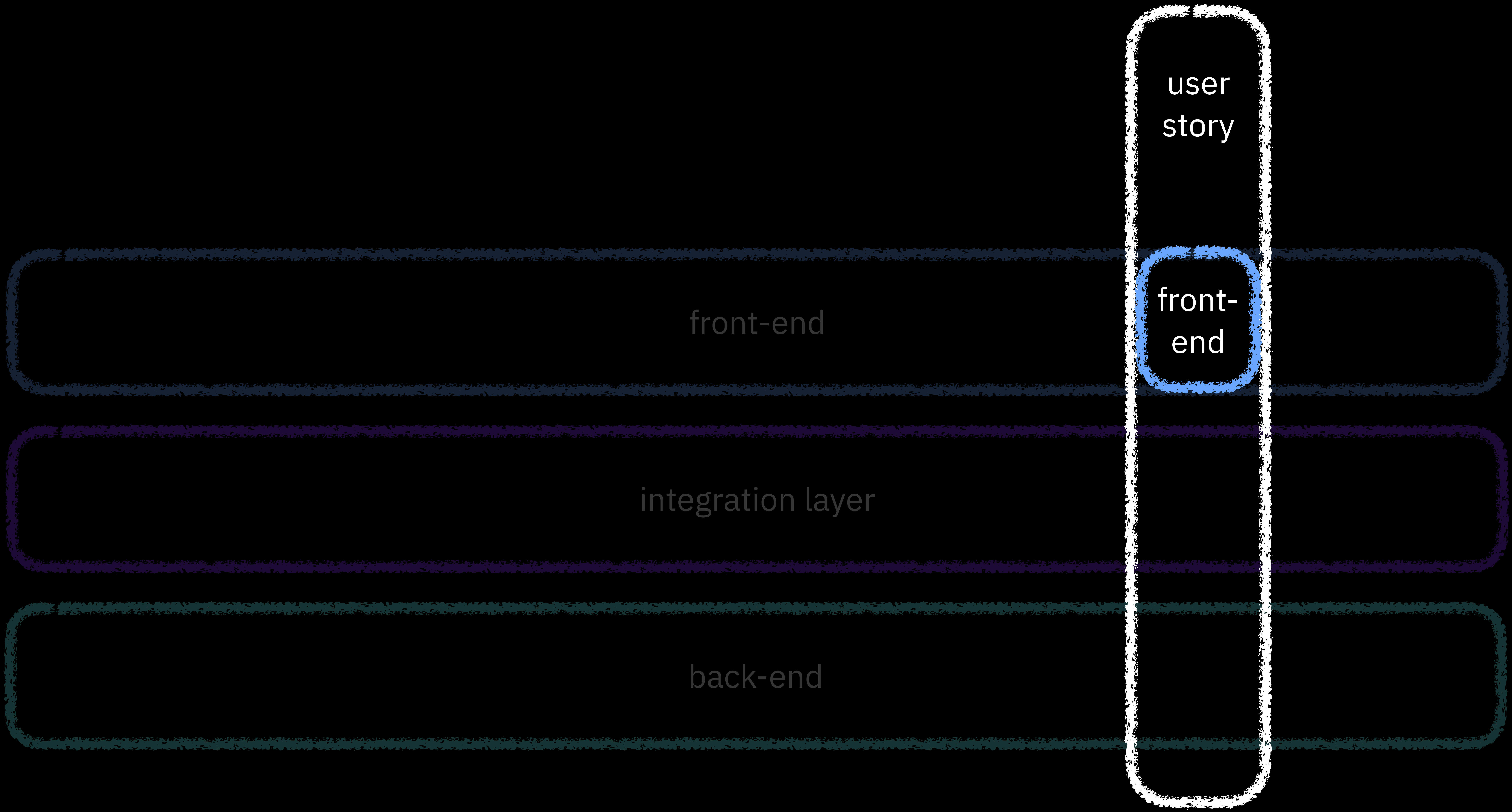
we deploy all our microservices at the same time.”

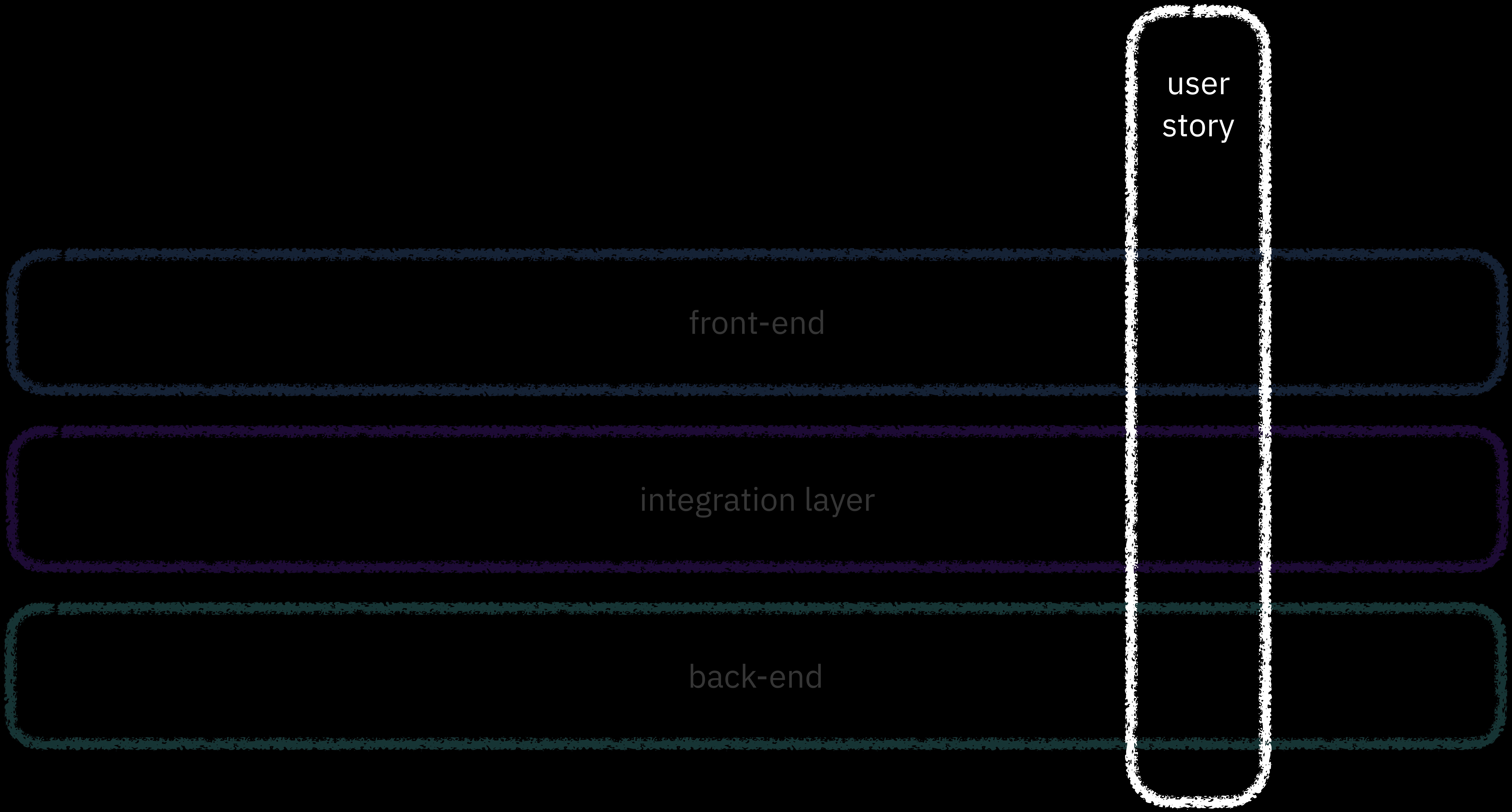
“it **looks** like it’s complete … but nothing works if you click on it.”

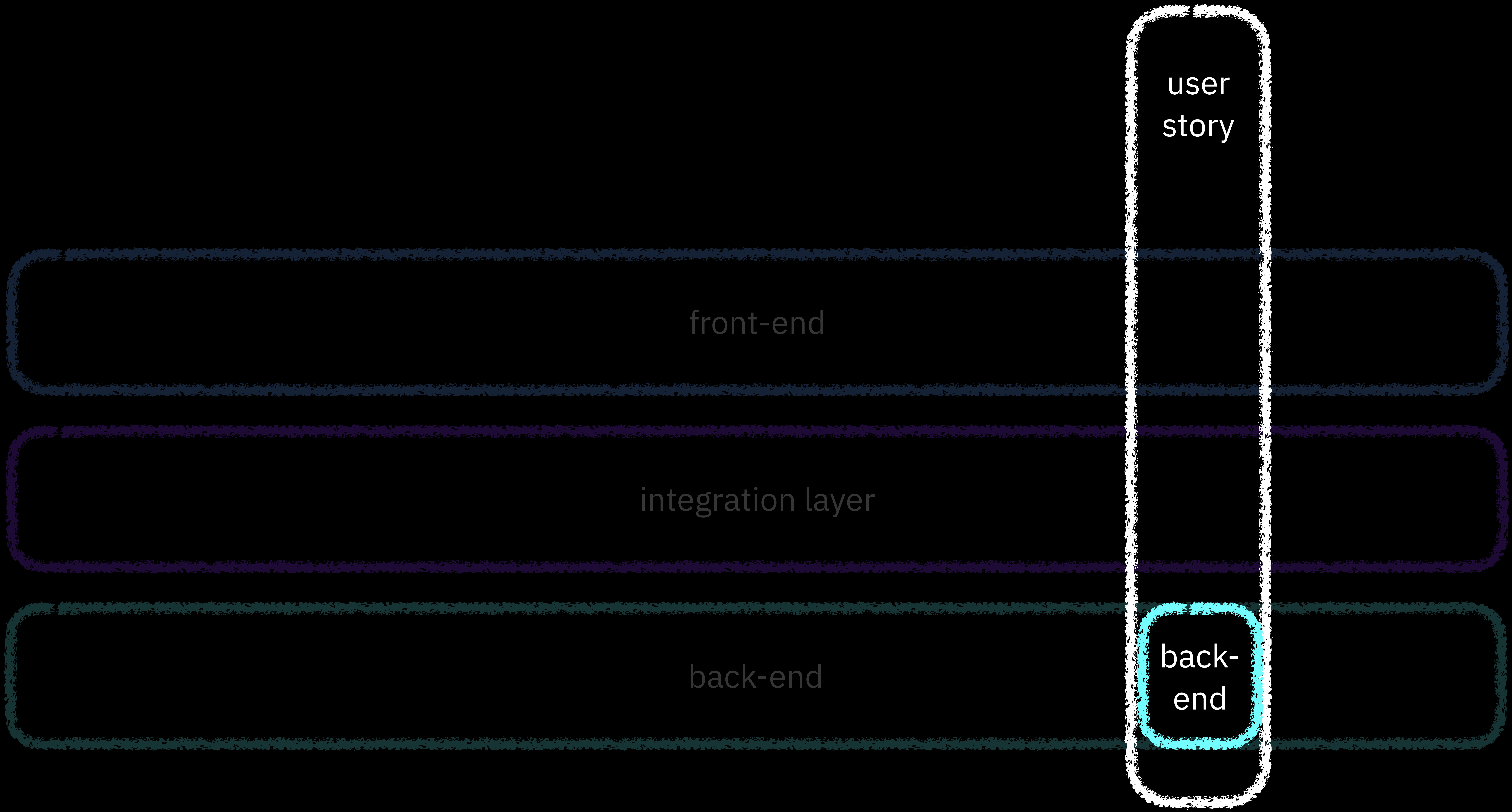

front-end
integration layer
back-end

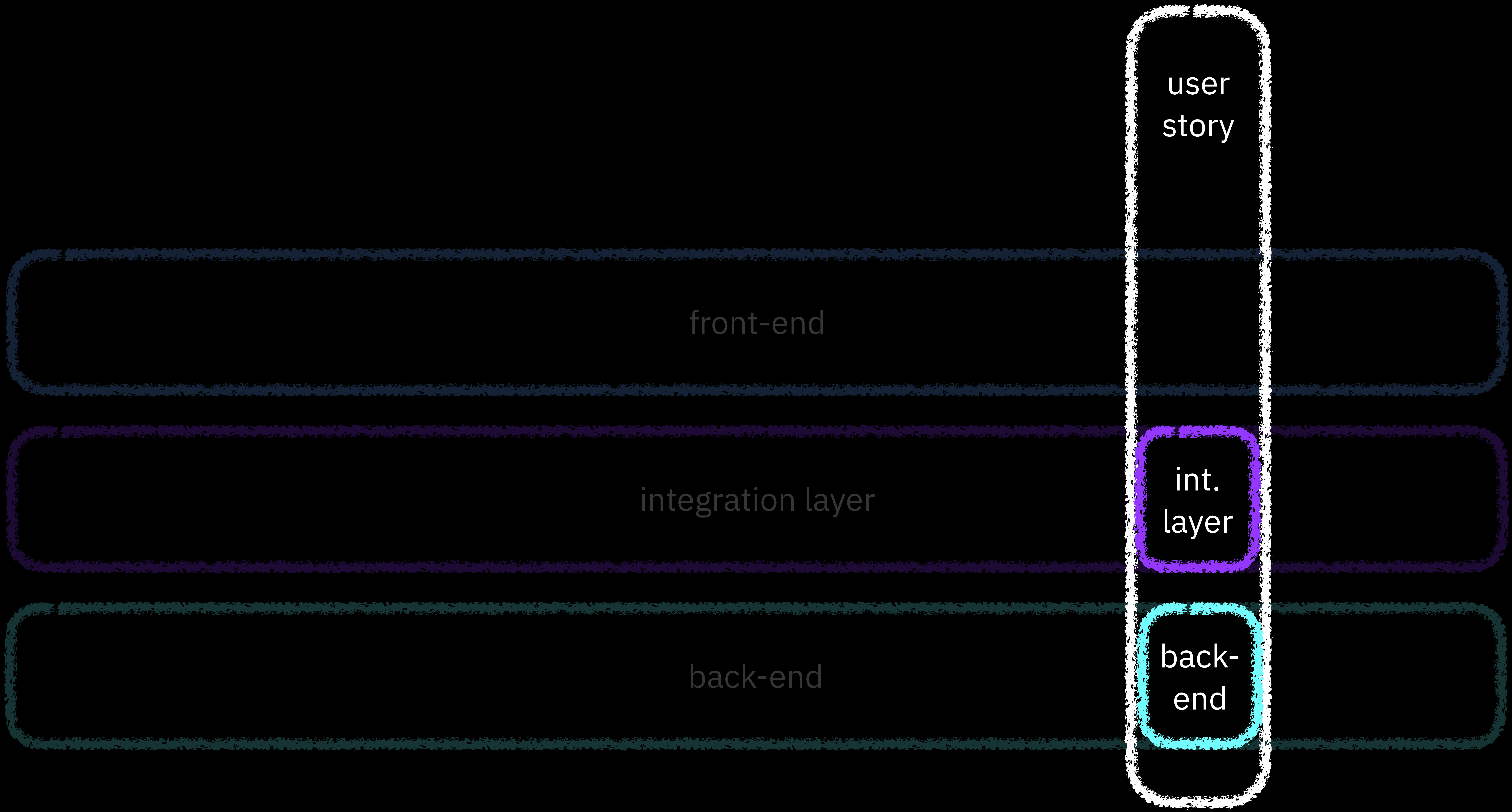

user story
front-end
integration layer
back-end


user story
front-end
integration layer
back-end
front-end
int. layer
back-end

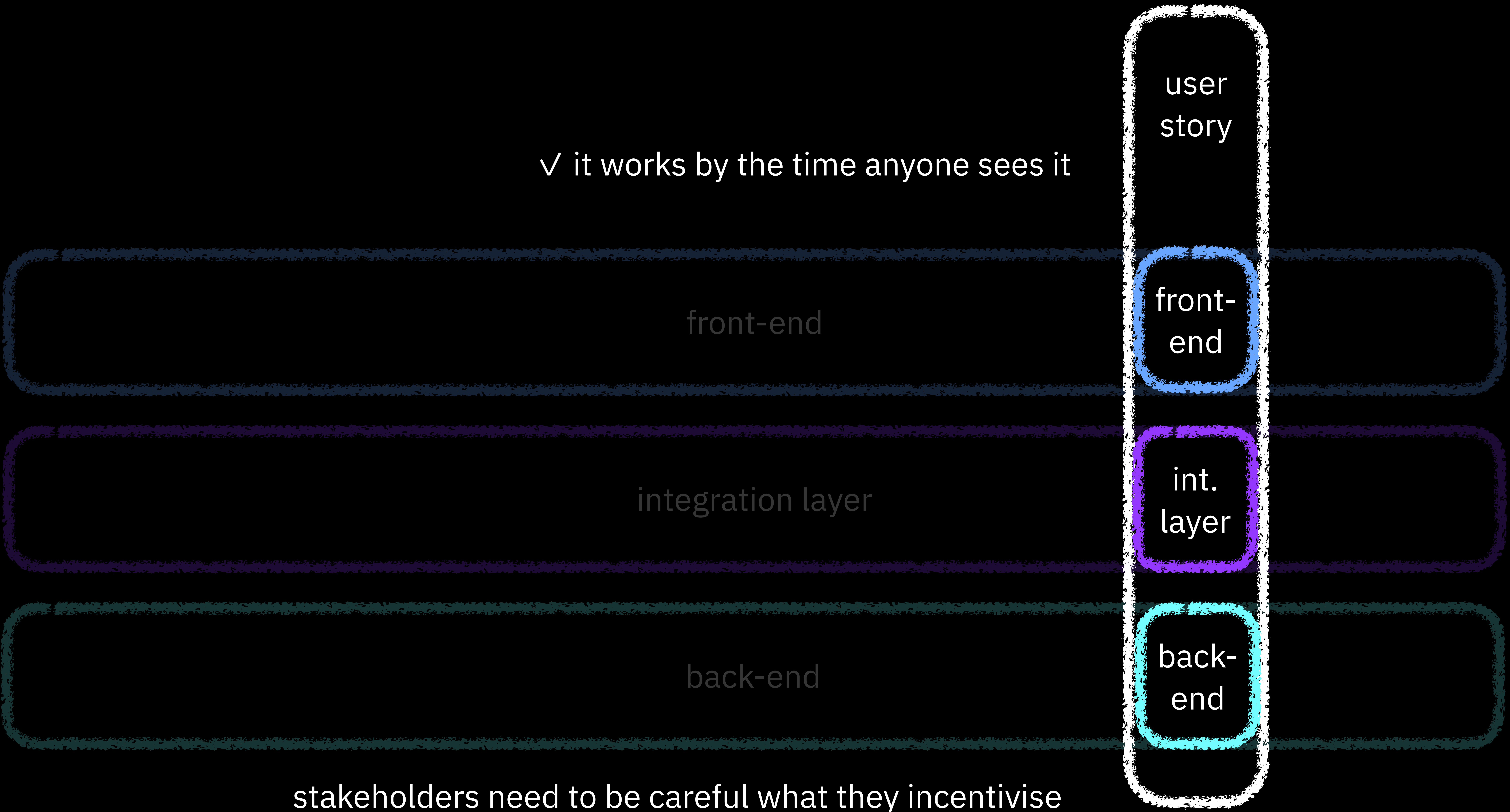

user story
front-end
front-end
integration layer
back-end


user story
front-end
integration layer
back-end


user story
front-end
integration layer
back-end
back-end


user story
front-end
integration layer
int. layer
back-end
back-end

√ it works by the time anyone sees it



stakeholders need to be careful what they incentivise

vertical slices

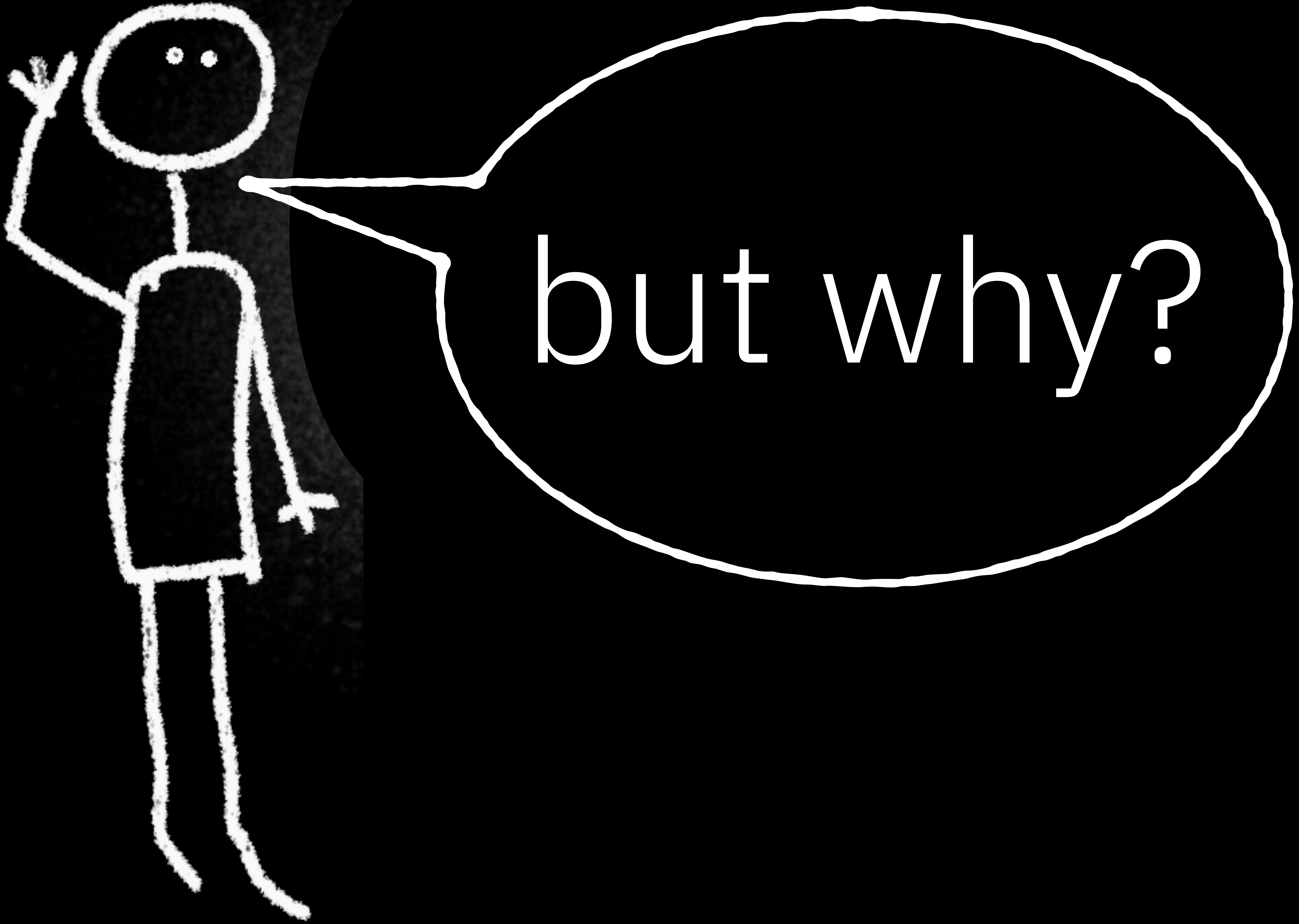
back-out development

back-first development
development

deferred wiring

feature flags

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


but why?

“users won’t find it compelling enough if we release now”

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

- Reid Hoffman

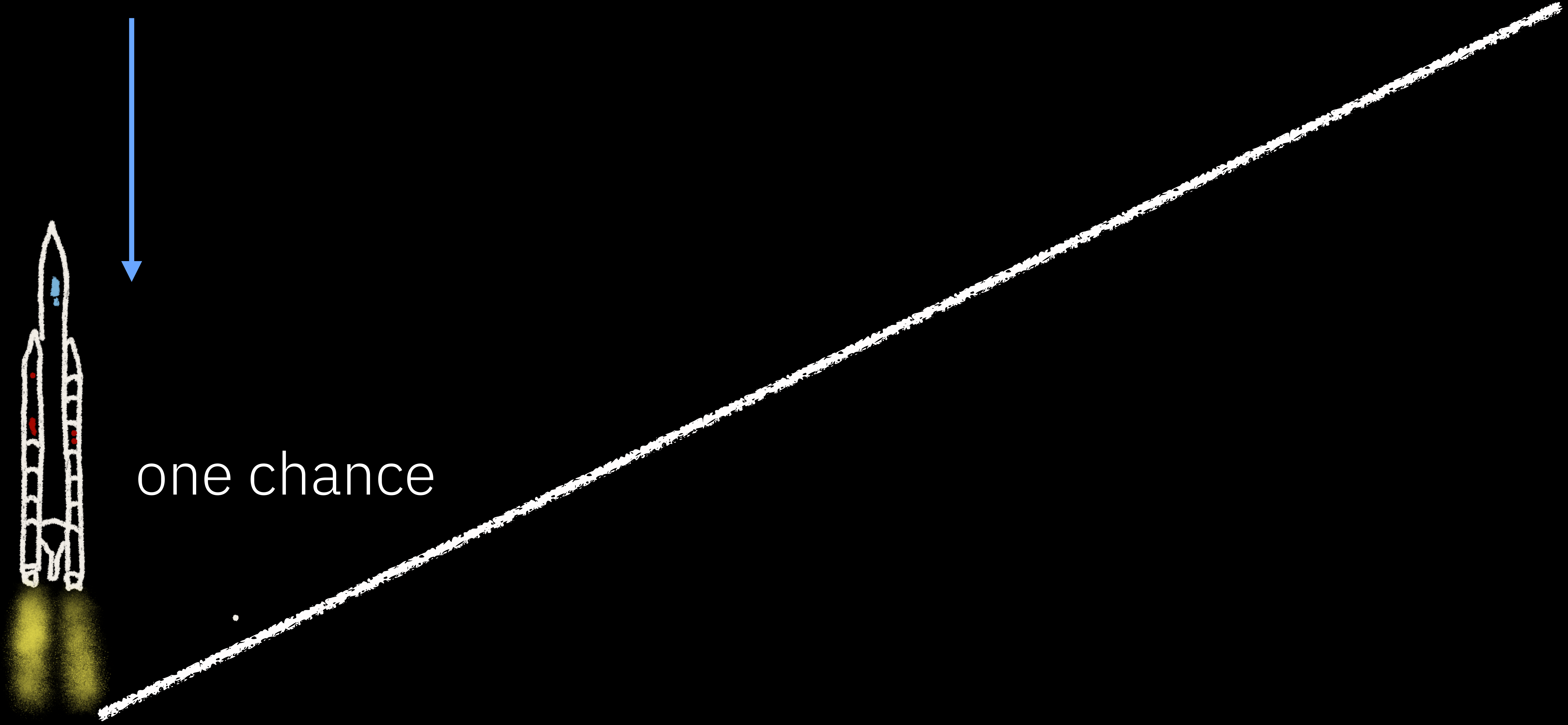
lean

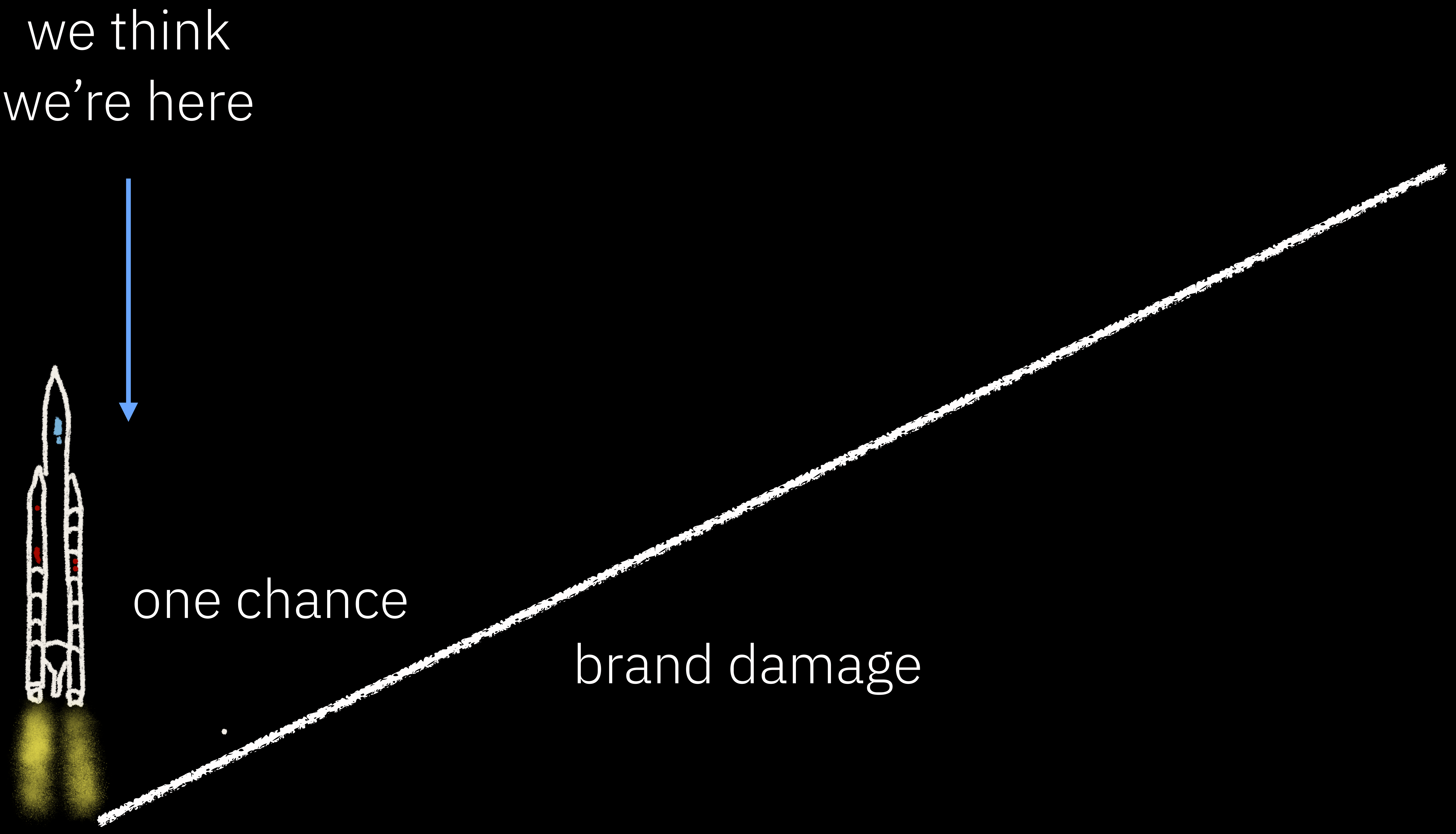
“we only get one chance
to get it right”

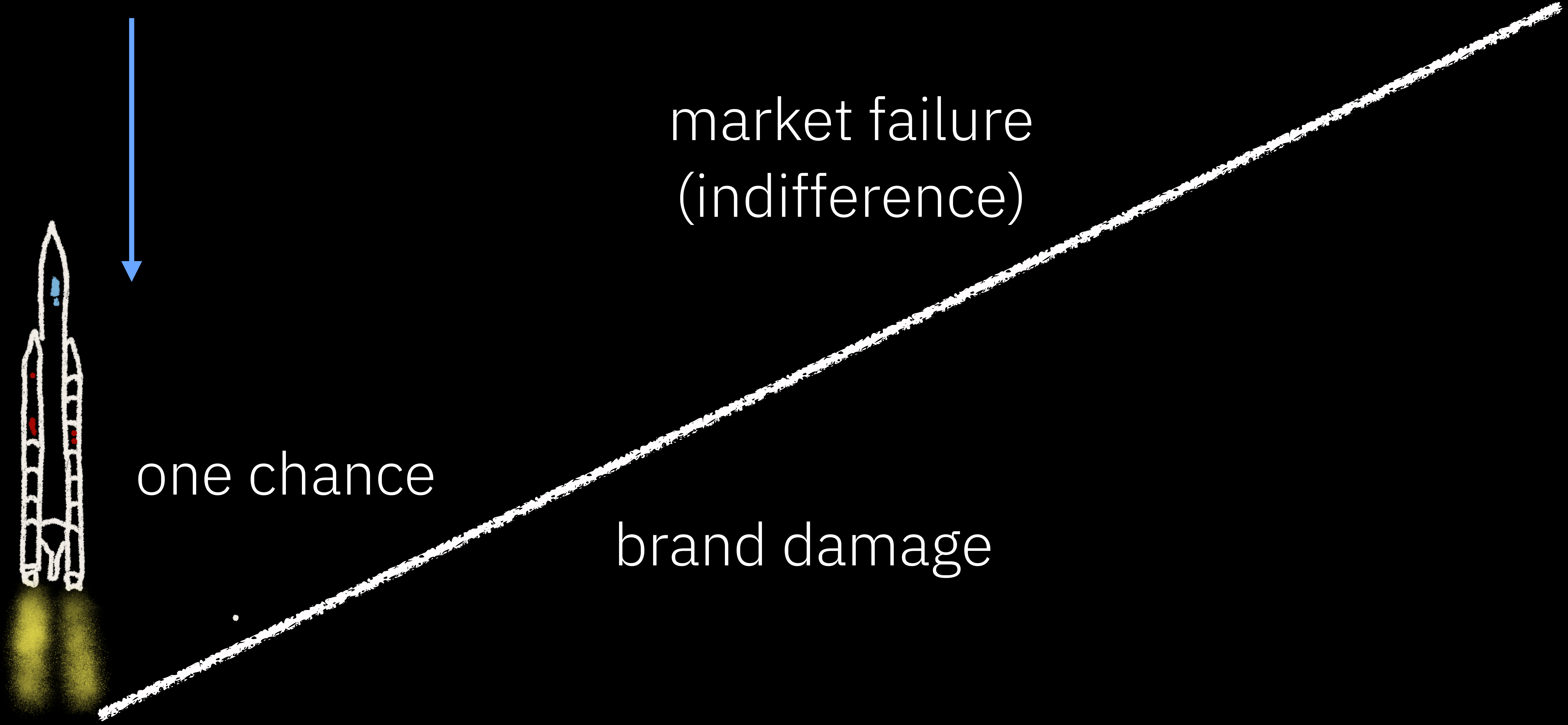
the ariadne failed in 36 seconds

you can’t a/b test a
$370 million rocket

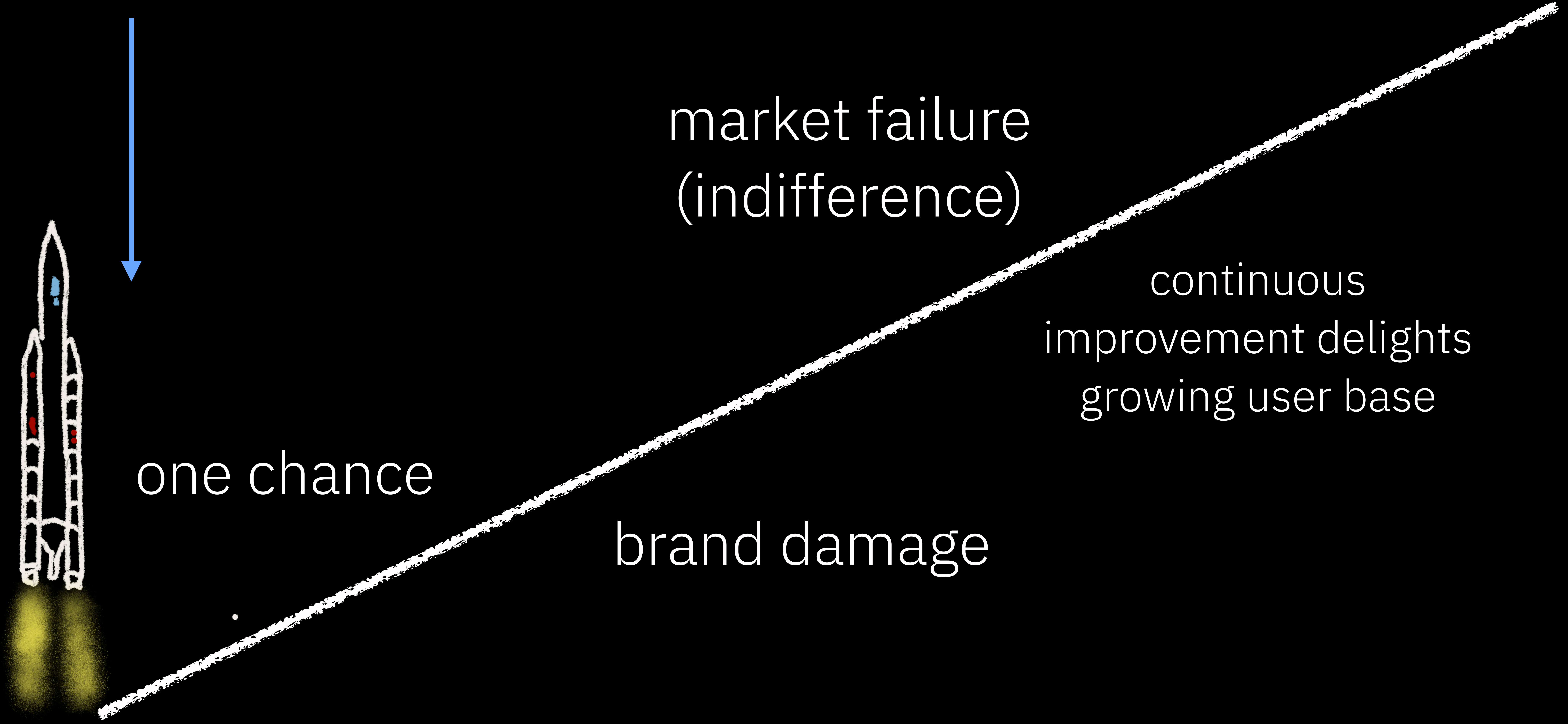
we think
we're here

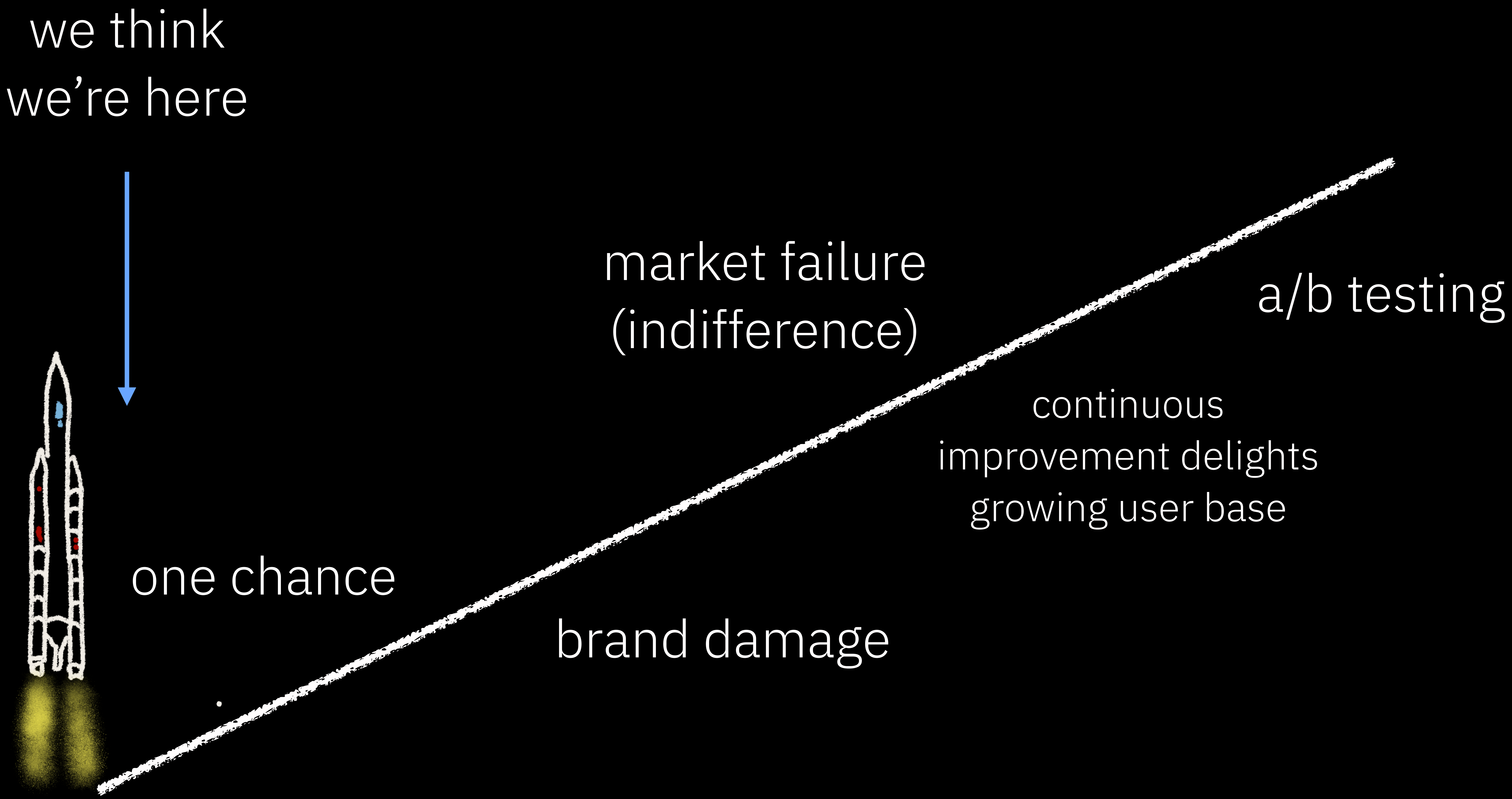


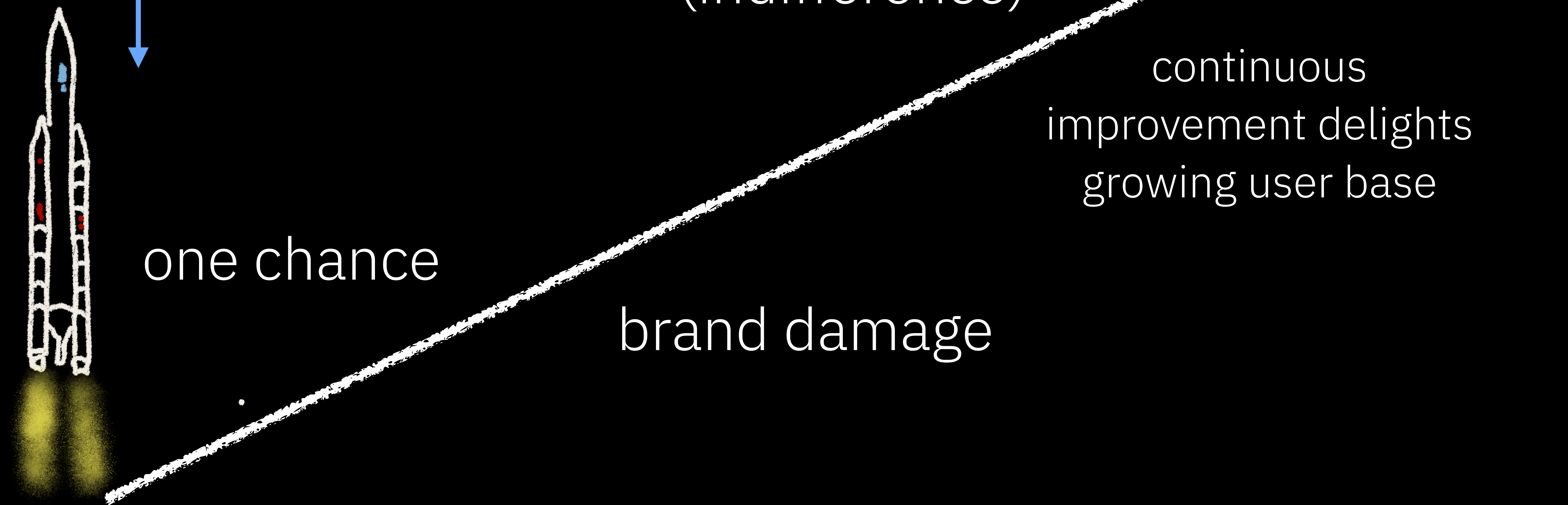

we think
we’re here
one chance
brand damage


we think
we're here
one chance
market failure
(indifference)
brand damage

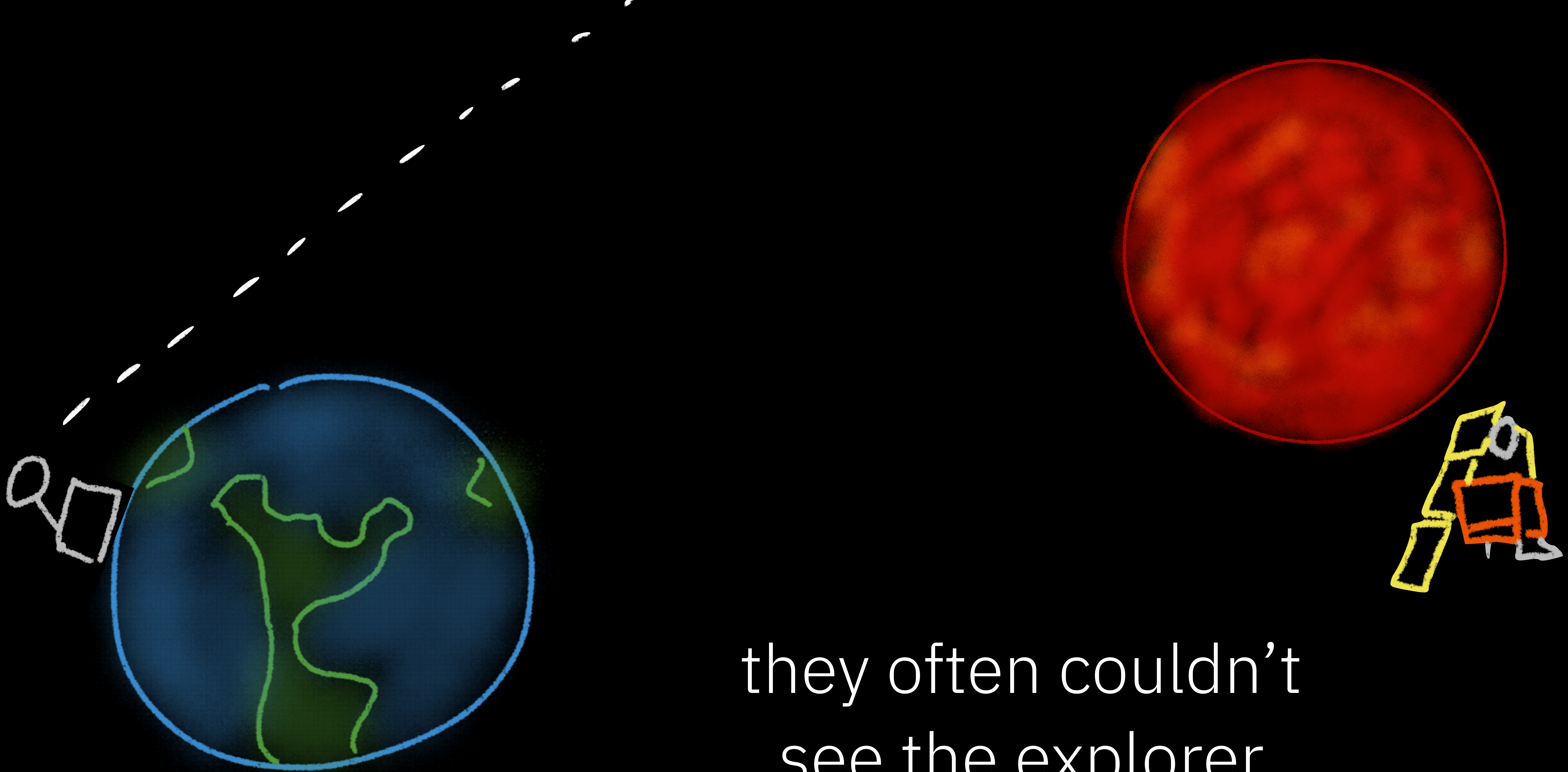
we think
we're here




we think
we're here
market failure
(indifference)
a/b testing
continuous
improvement delights
growing user base
one chance
brand damage


we think we’re here
could we be here?
one chance
brand damage
market failure (indifference)
continuous improvement delights growing user base
a/b testing

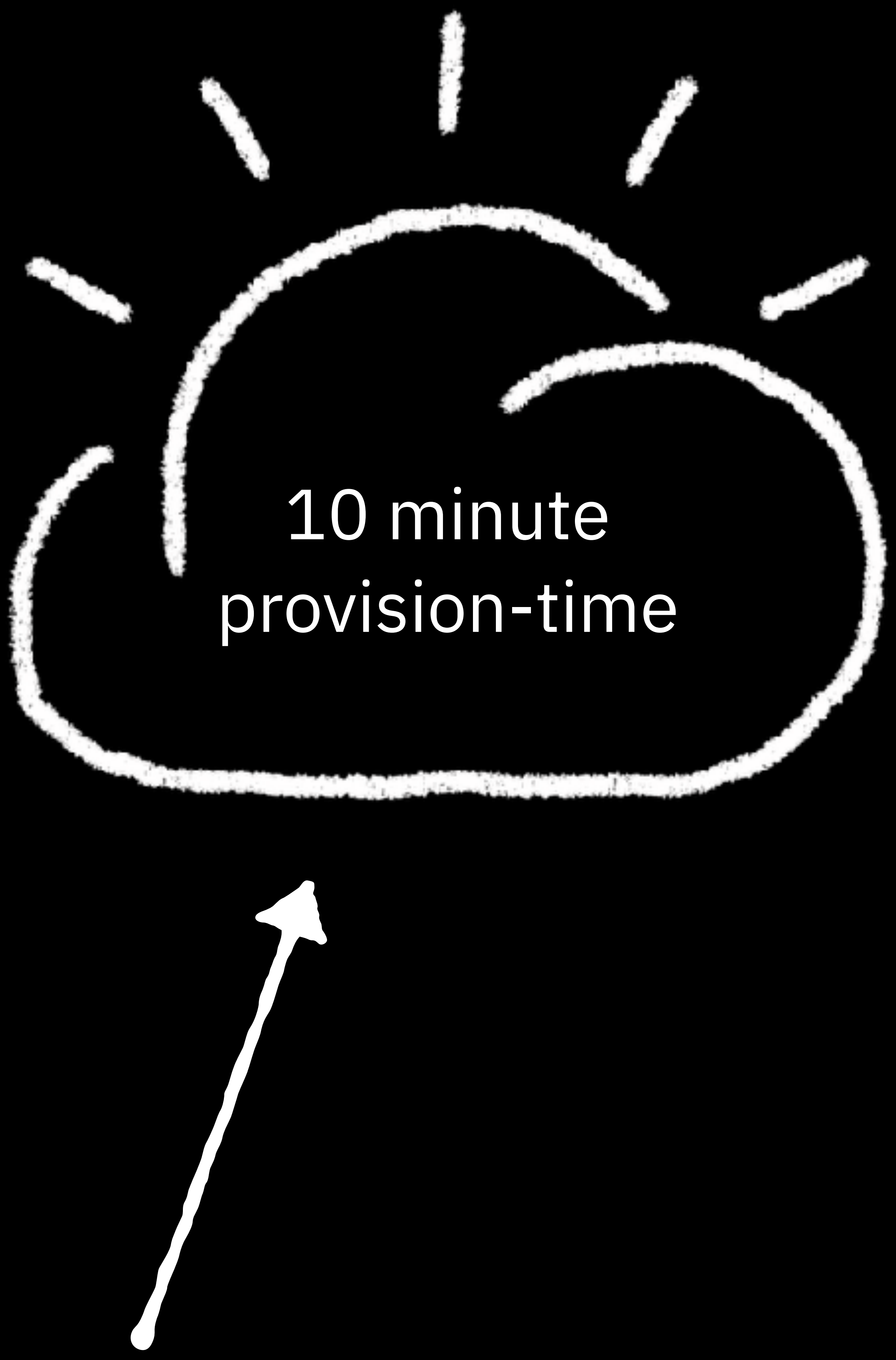
feedback is good engineering



they often couldn't see the explorer

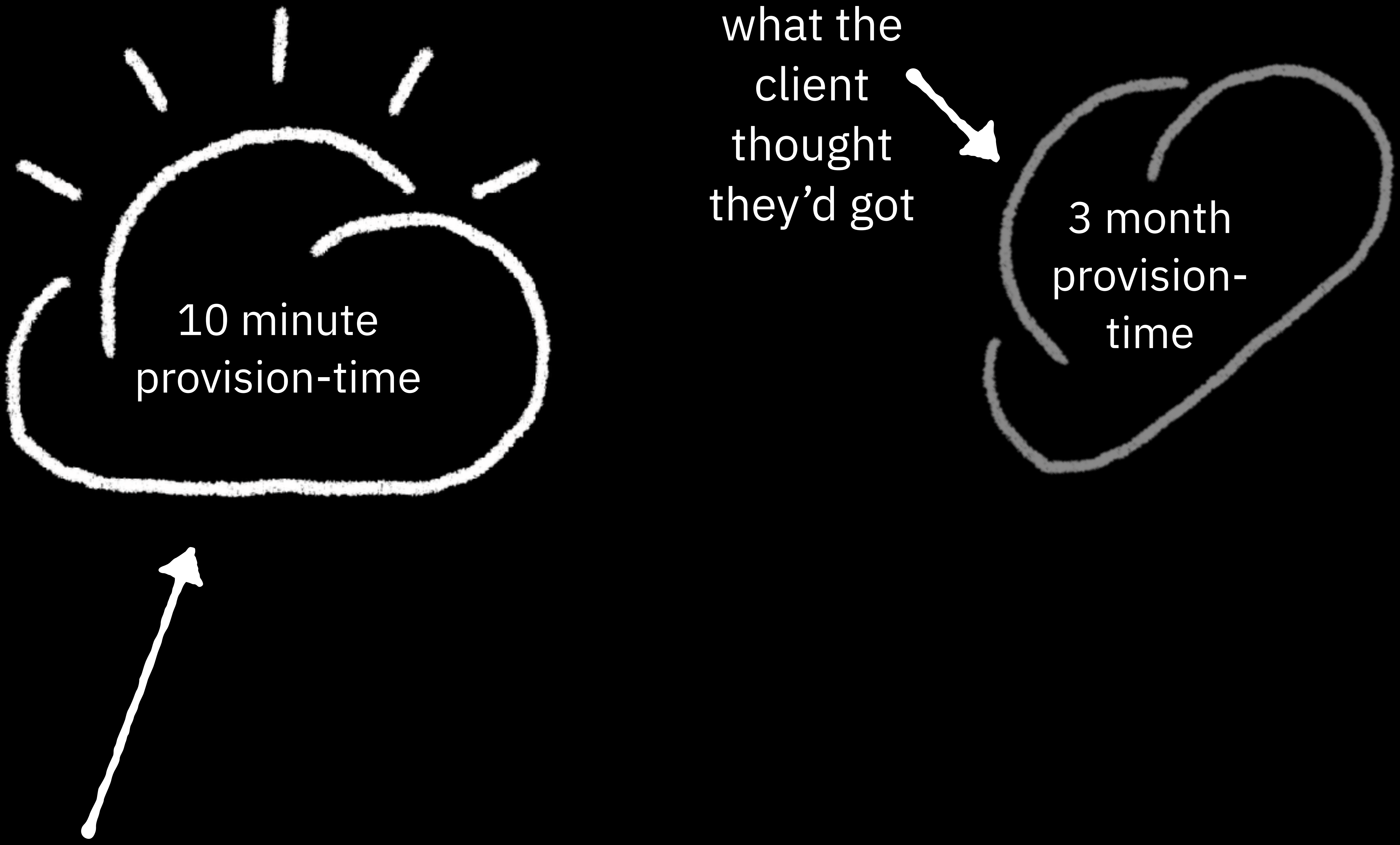
“but our change control process …”

“this provisioning software is broken”



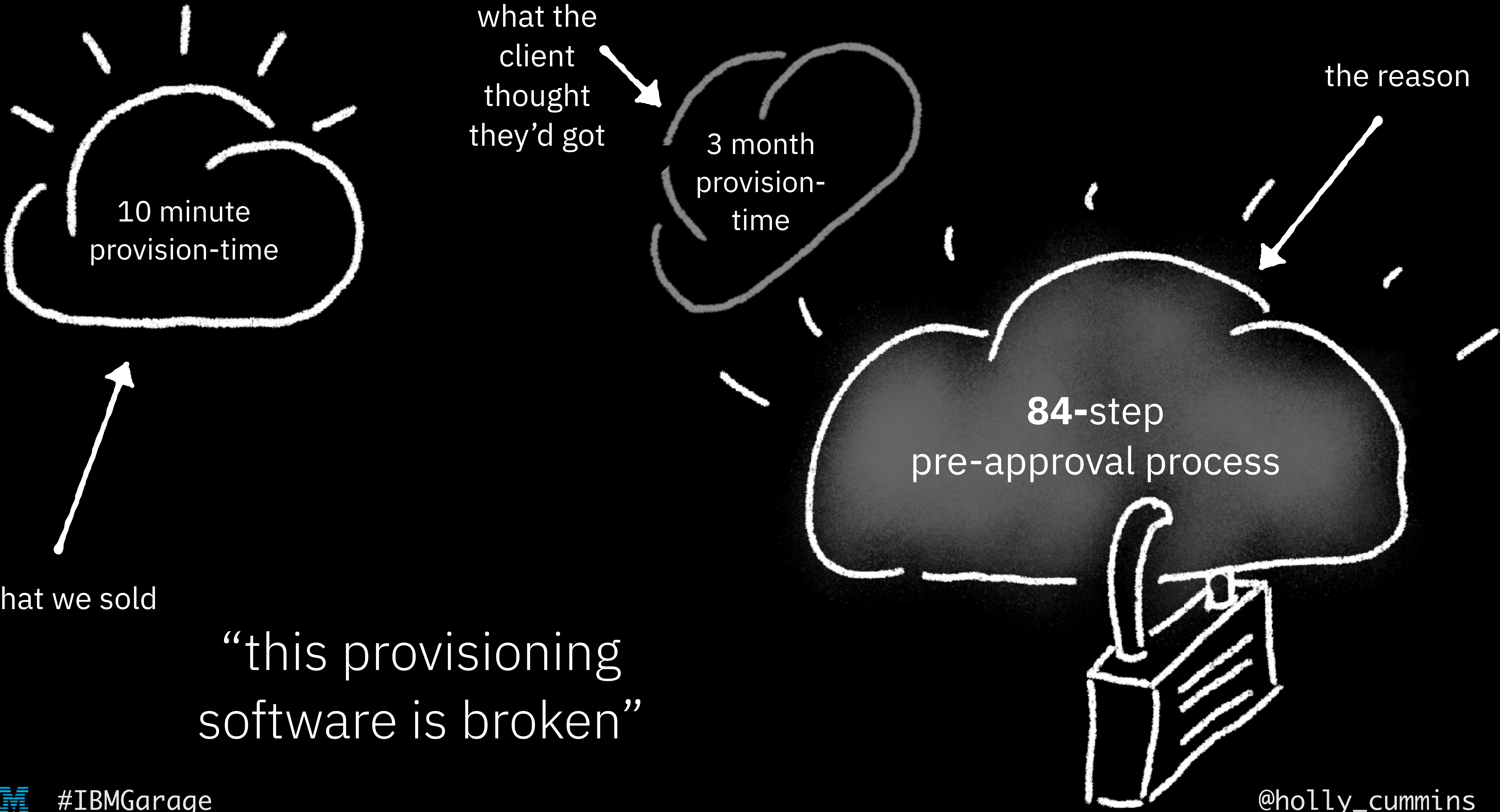
what we sold

“this provisioning software is broken”

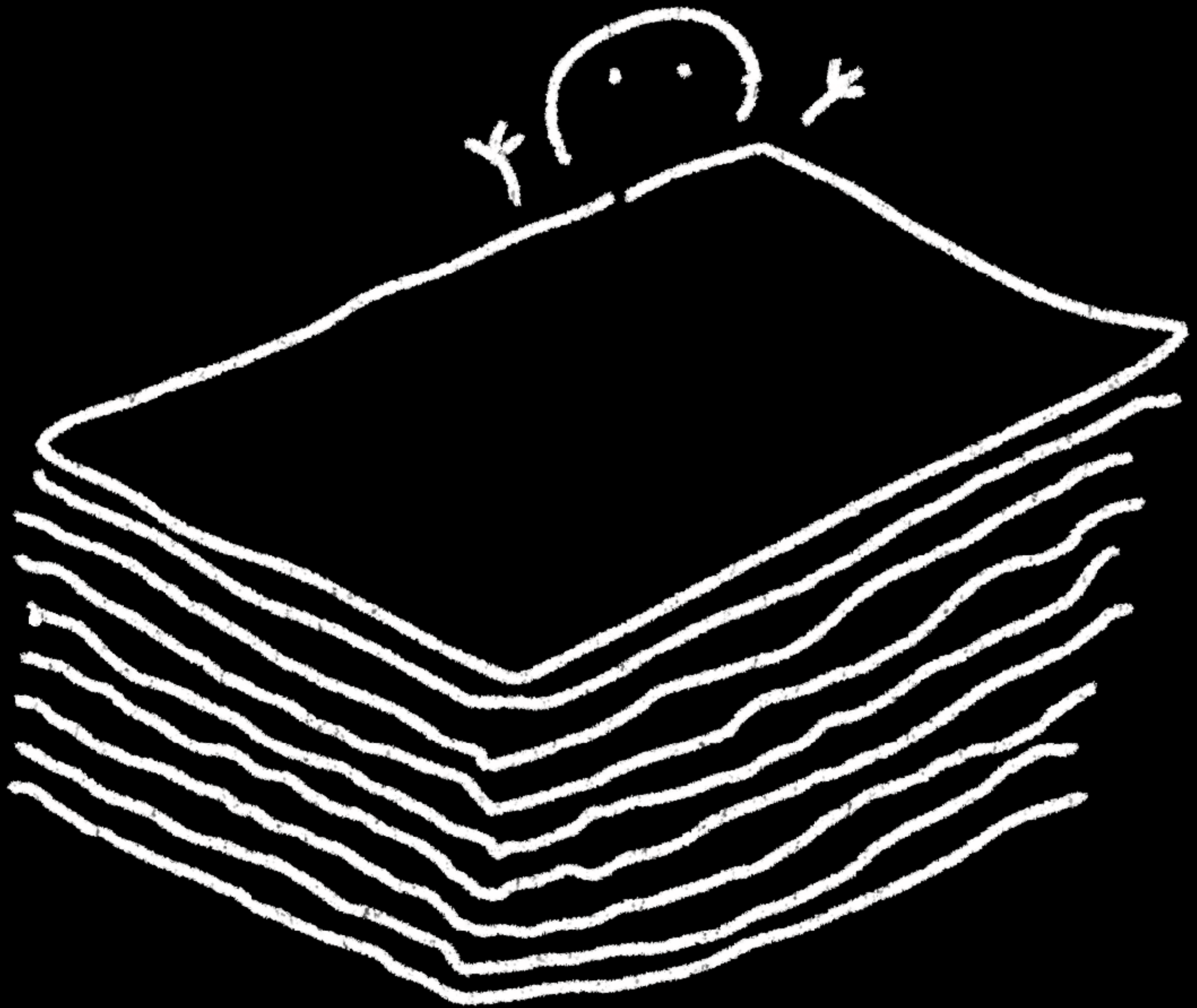


what we sold

“this provisioning software is broken”



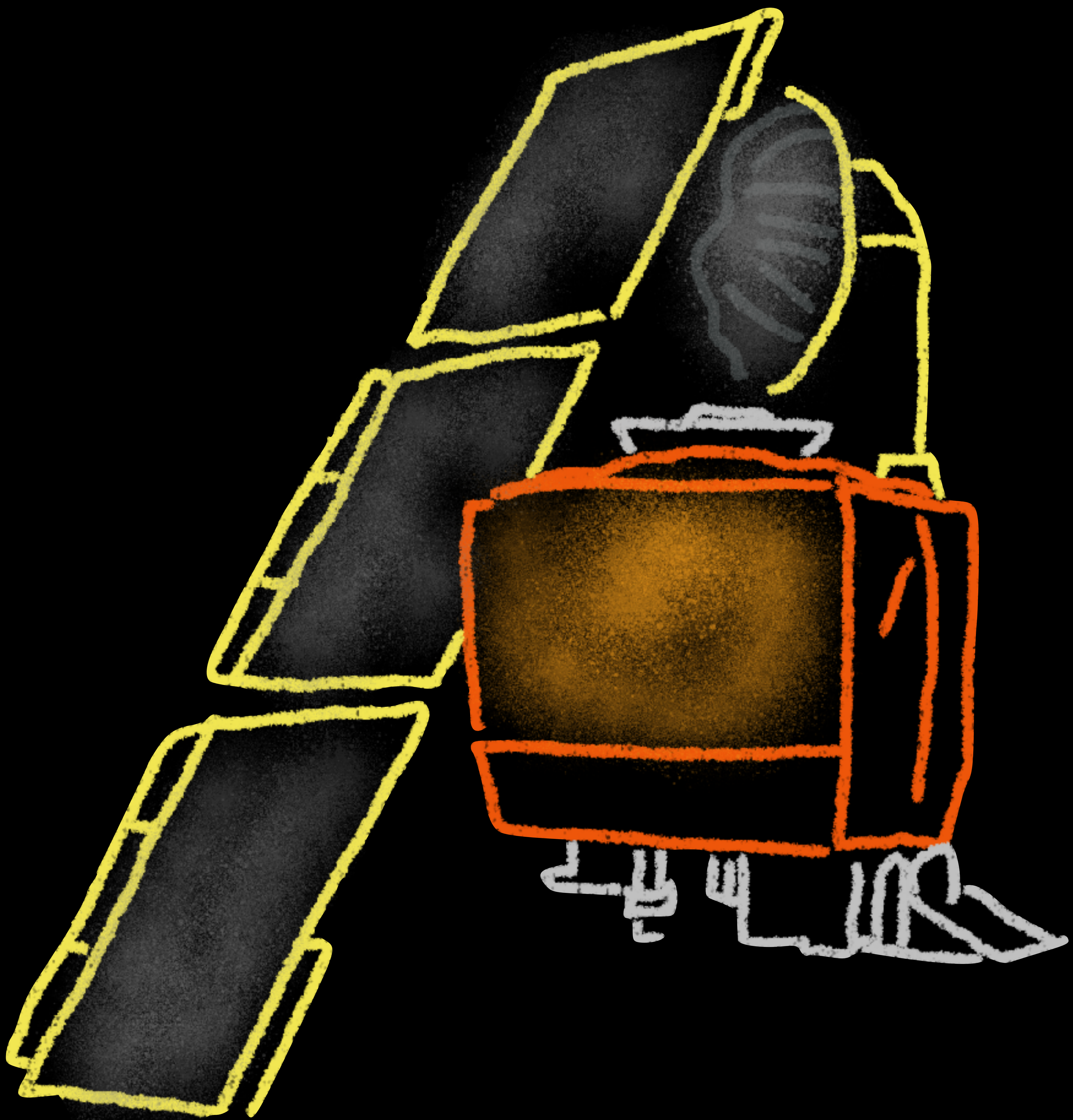
“this provisioning software is broken”



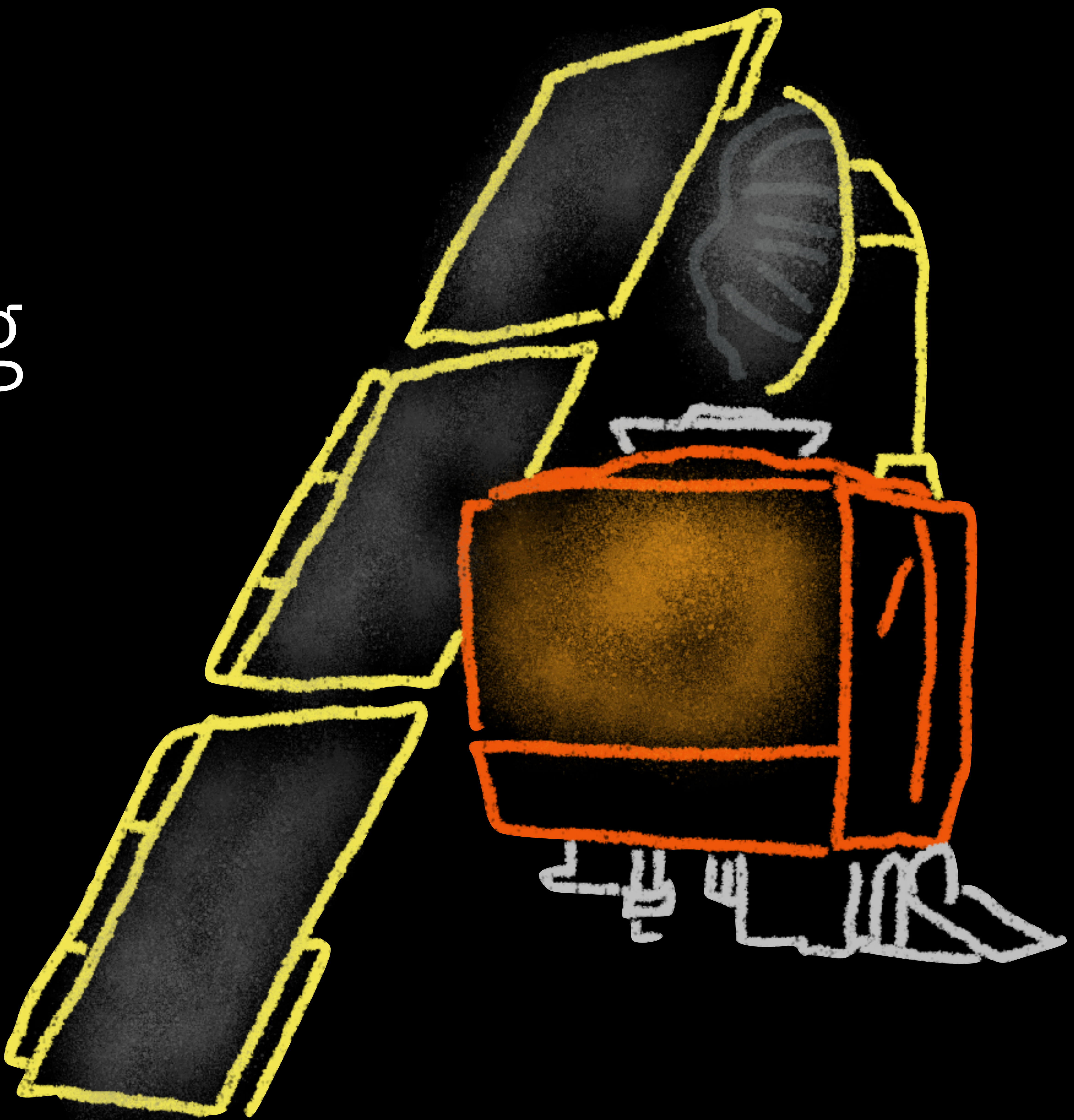
“we’ve scheduled the architecture board review for a month after the project ships”

@holly_cummins

does the process
add **value?**

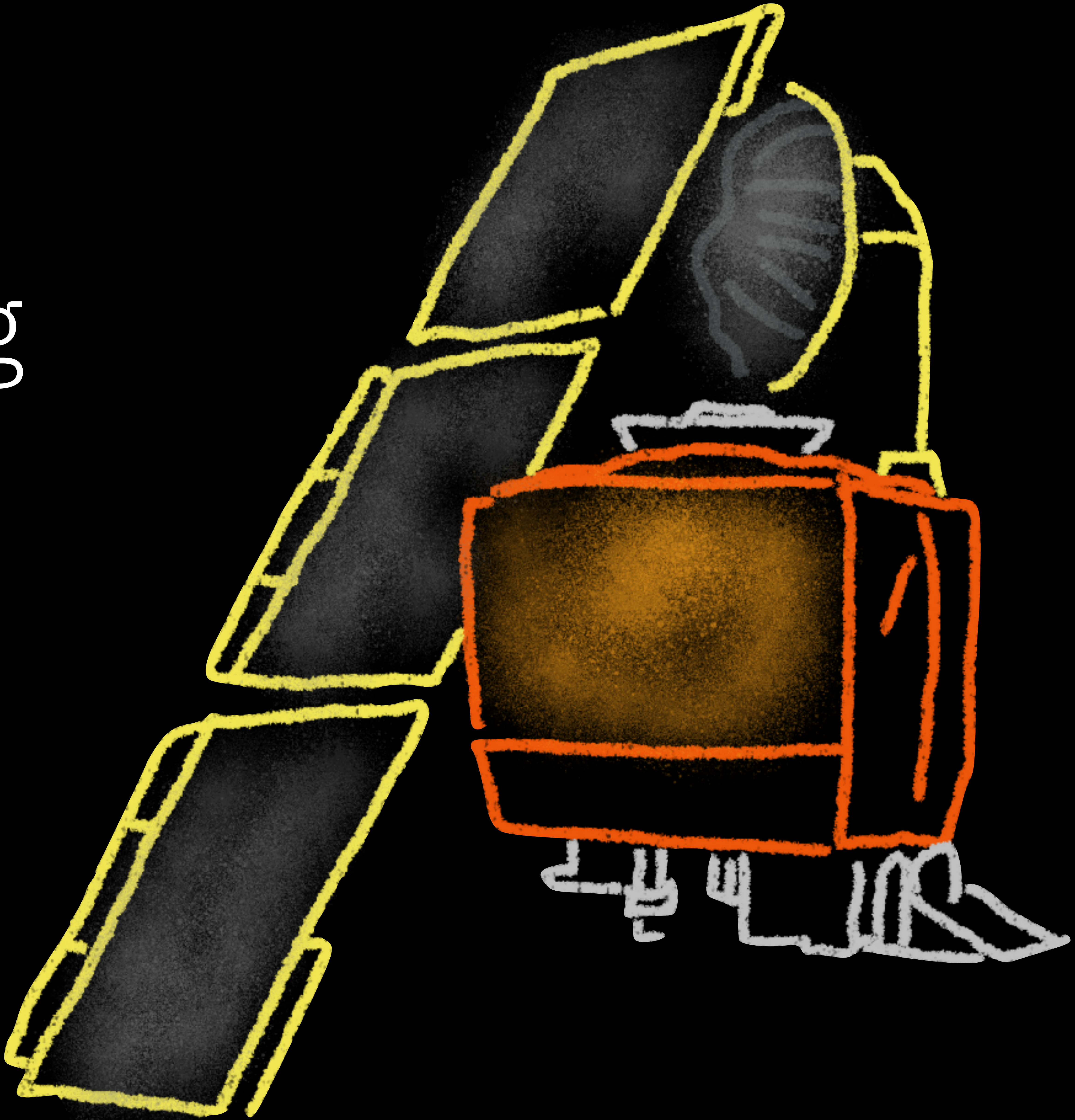


navigators warned something was wrong



navigators warned something was wrong

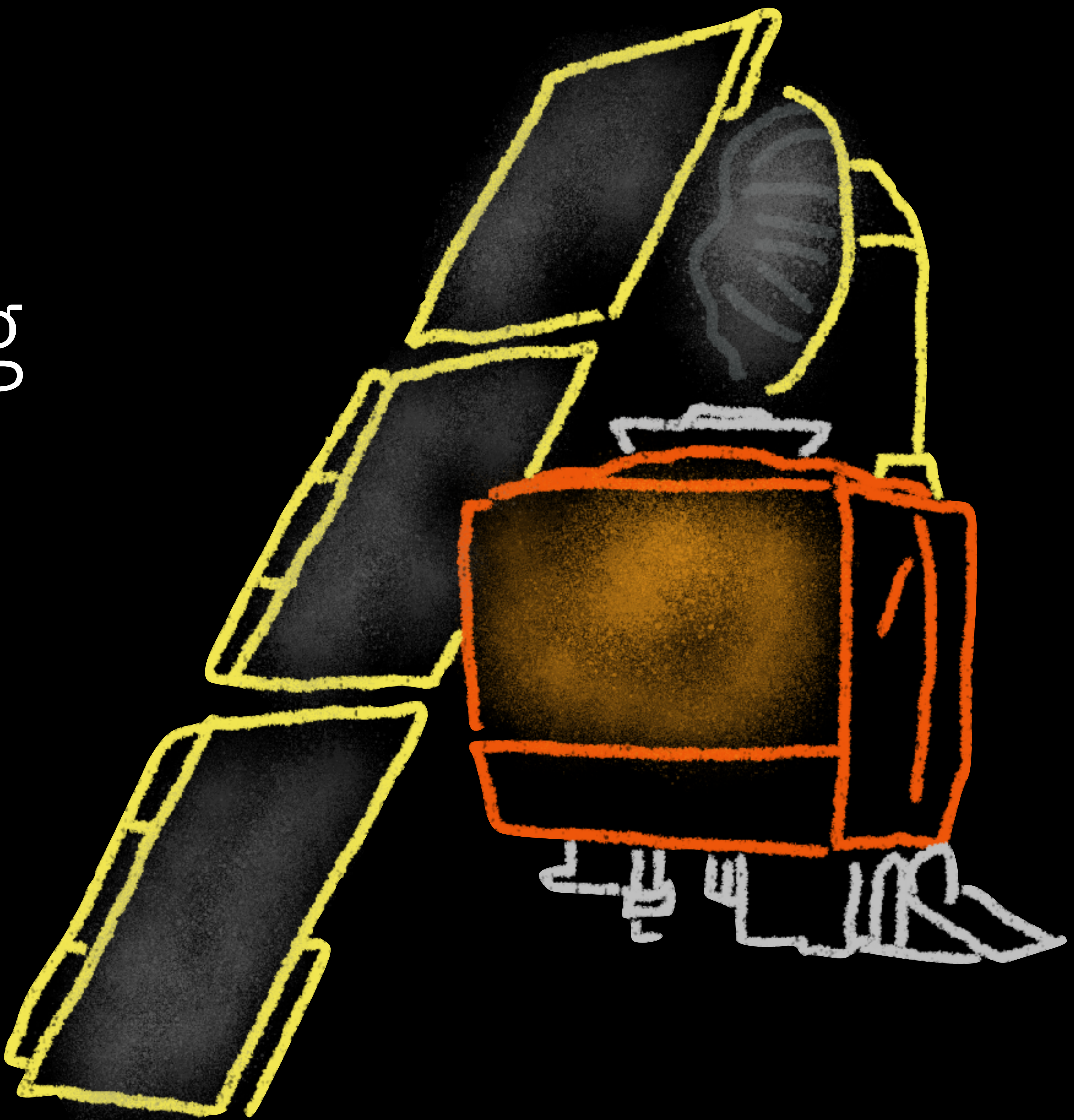
they didn't fill in the right form



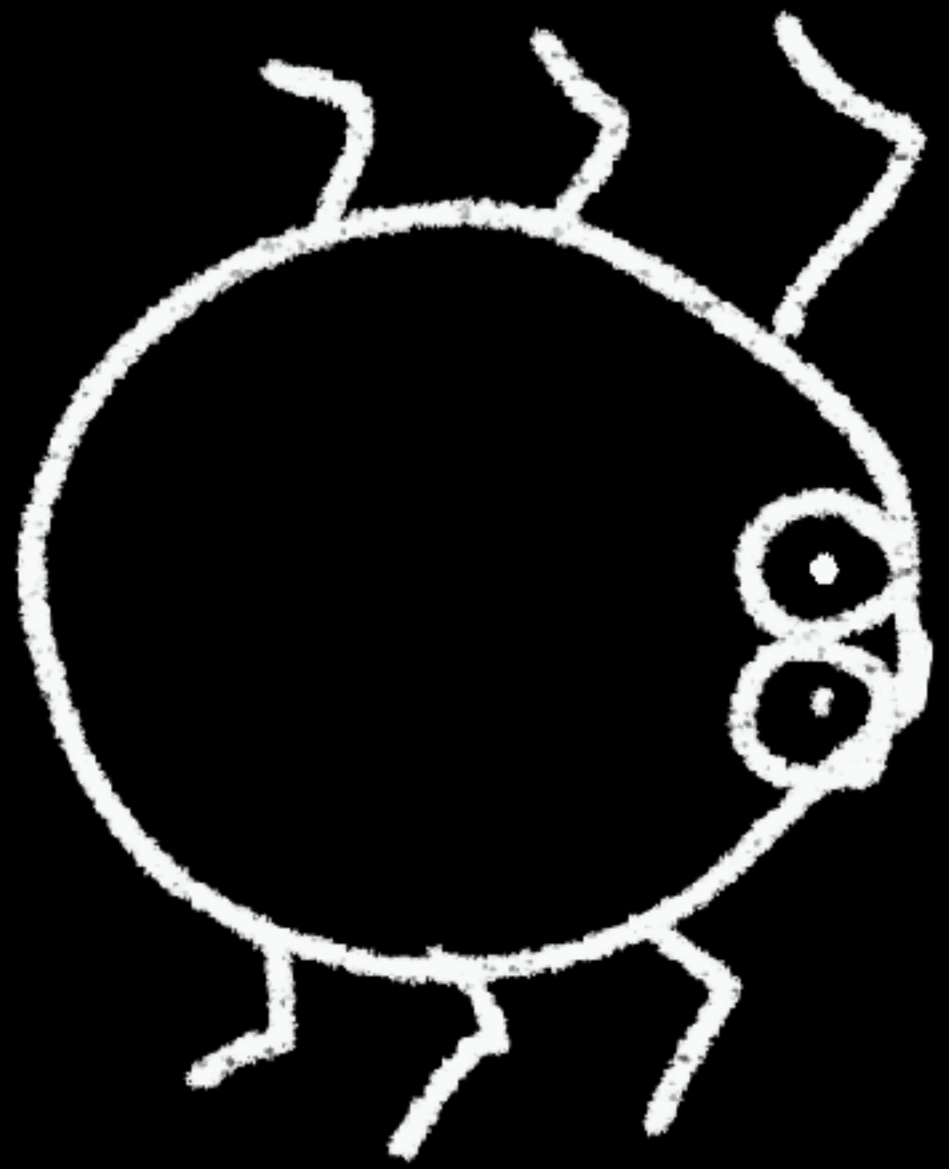
navigators warned something was wrong

they didn't fill in the right form

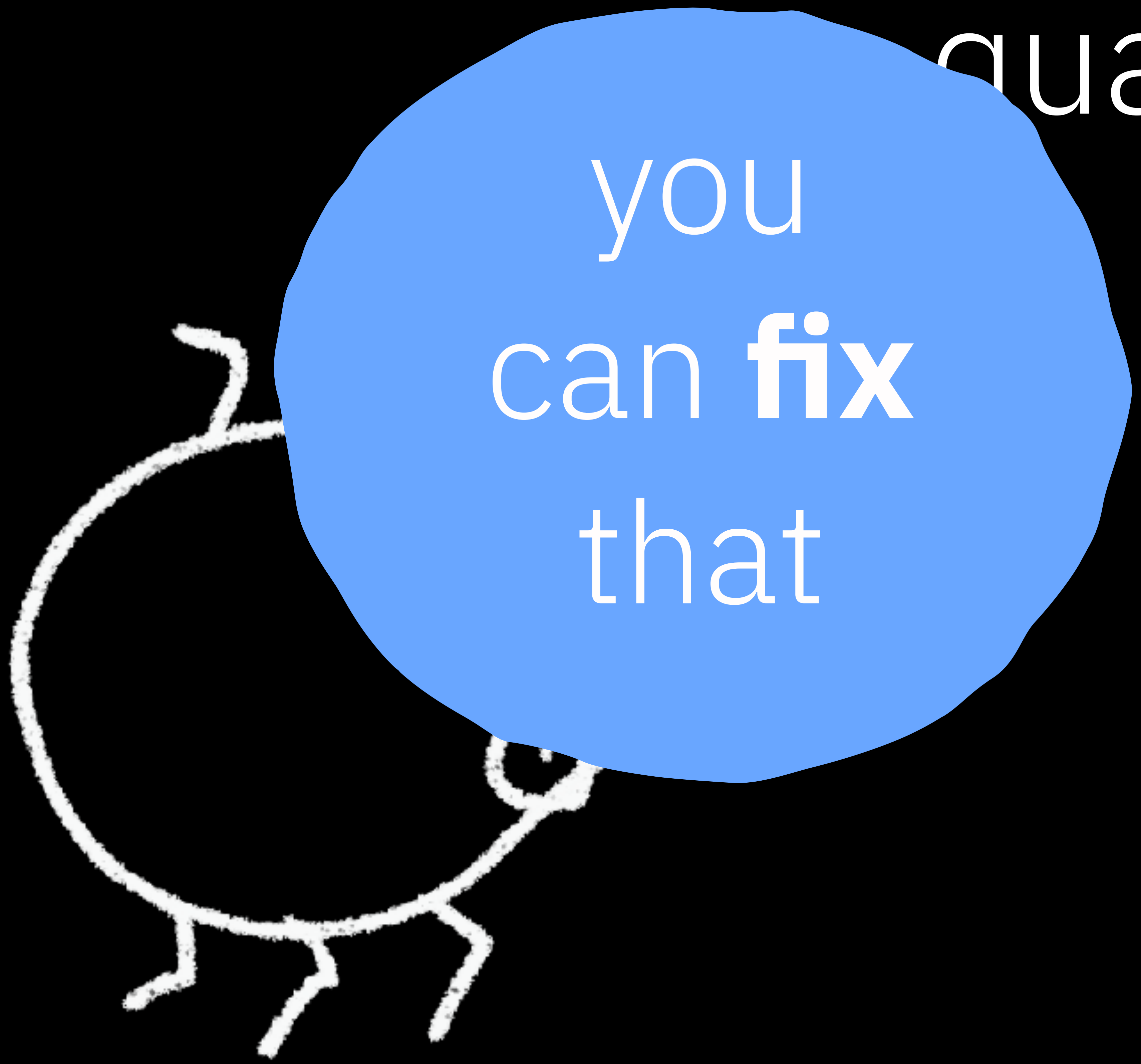
so nothing was done



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

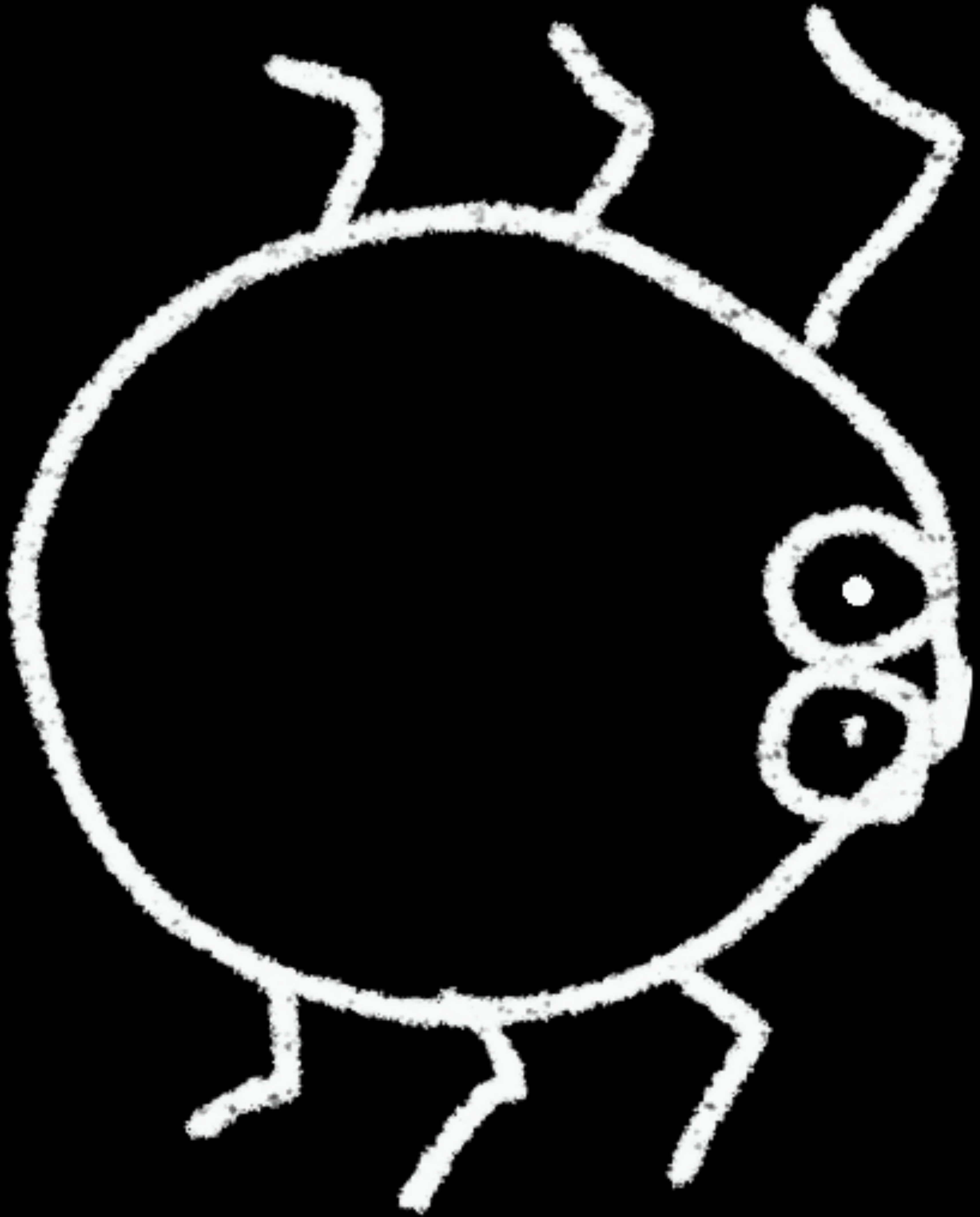


“we can’t ship until we have more confidence in the quality”



“this is the test team … who don’t have the skills to automate their tests.”

‘

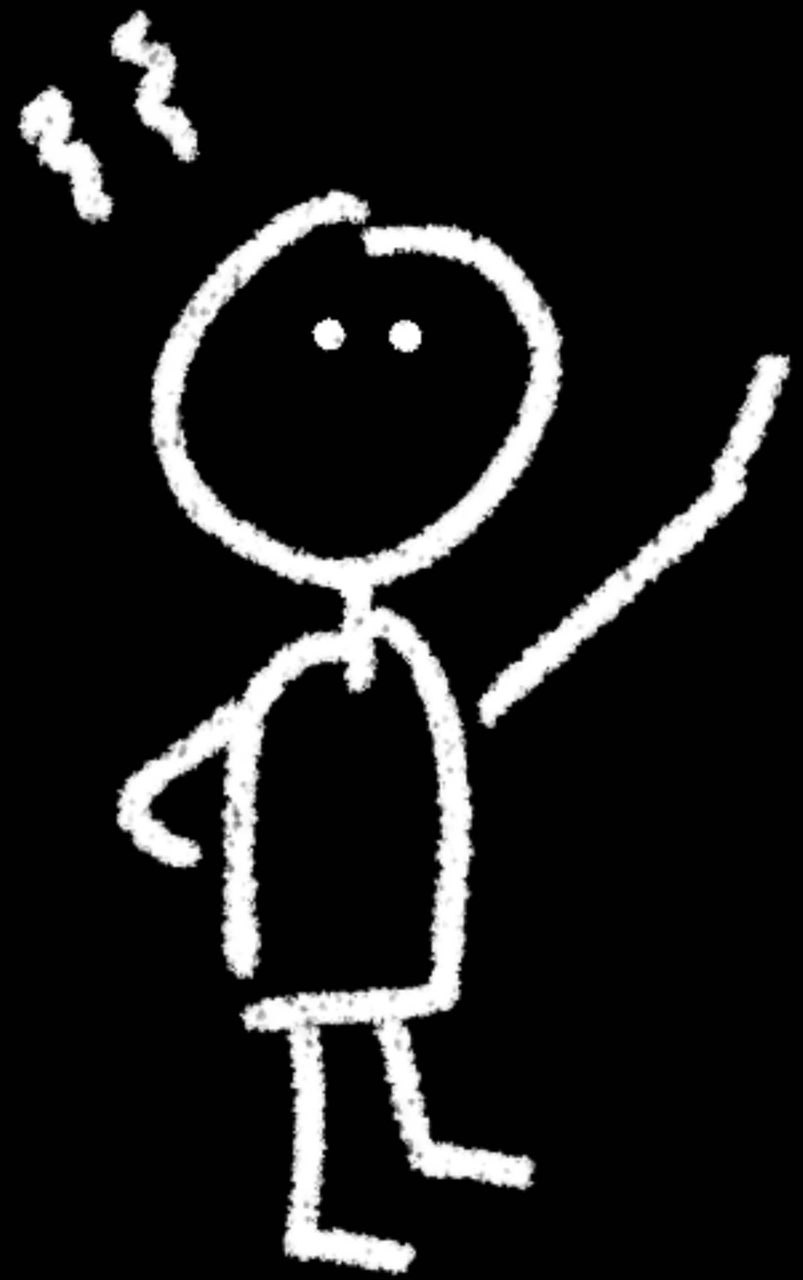


“our tests aren’t automated”

“we don’t know if our code currently works” “we don’t know if our code currently works” “it costs too much to release”

“it costs too much to release”

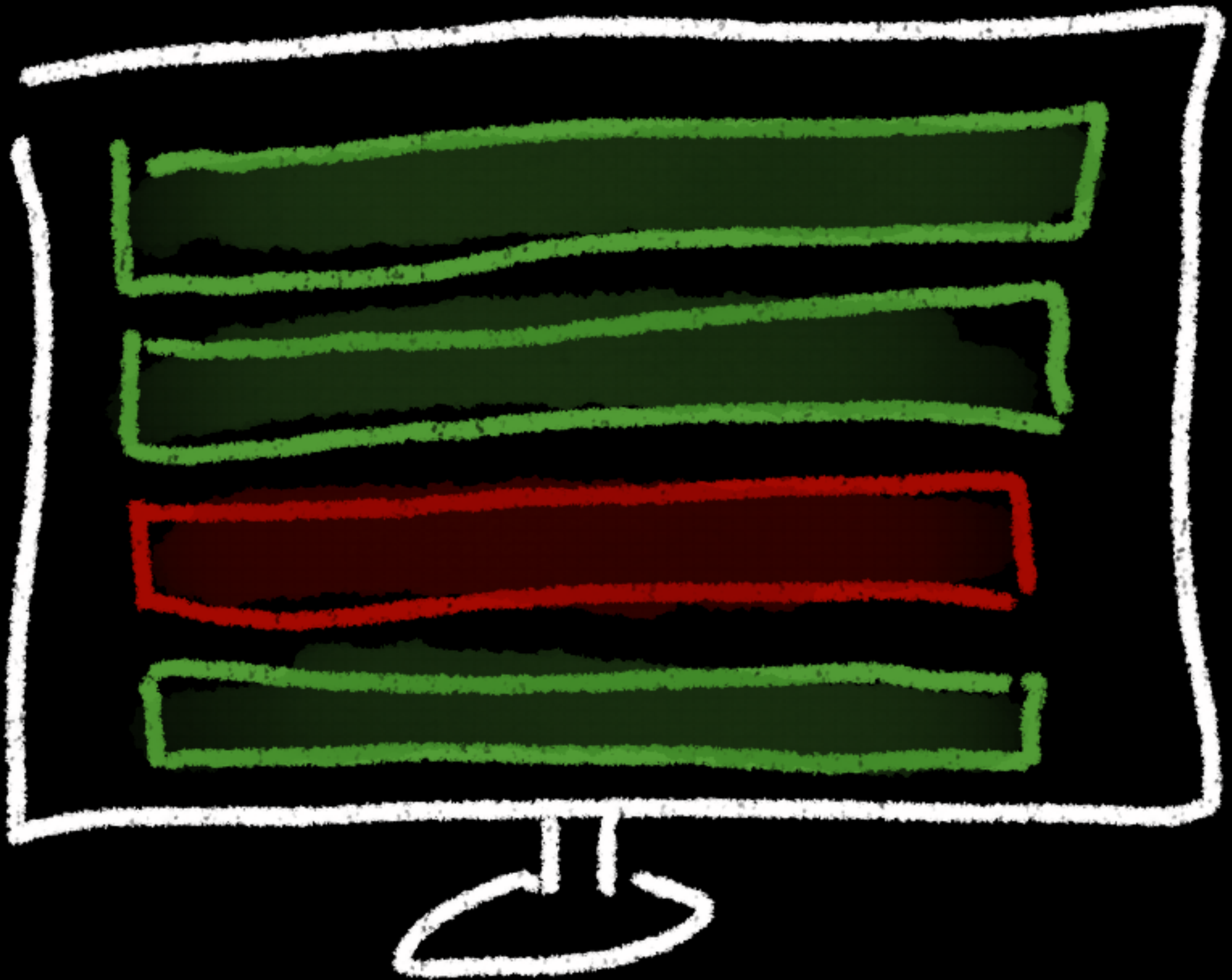
you
can **fix**
that



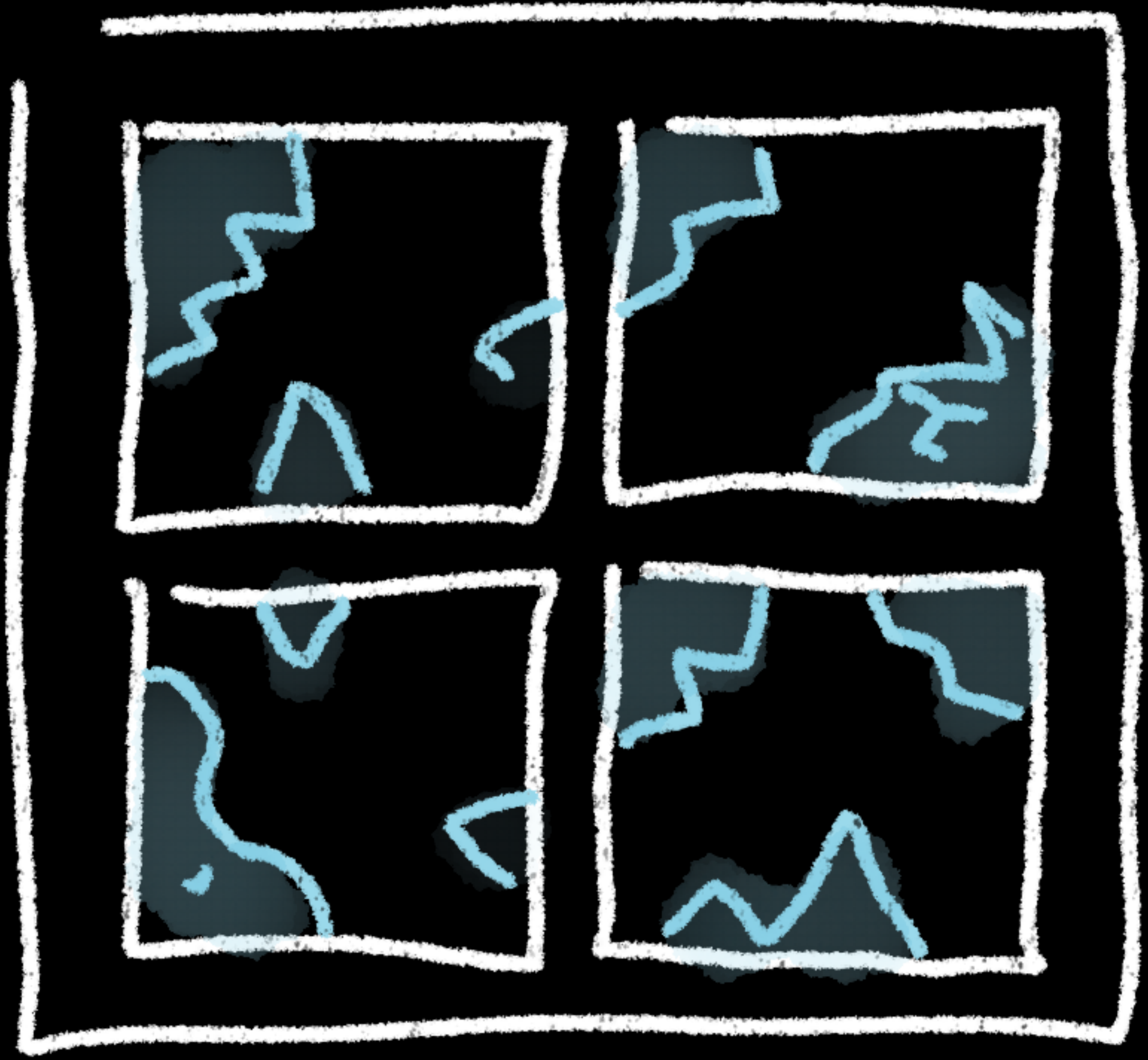
not a good CI/CD indicator

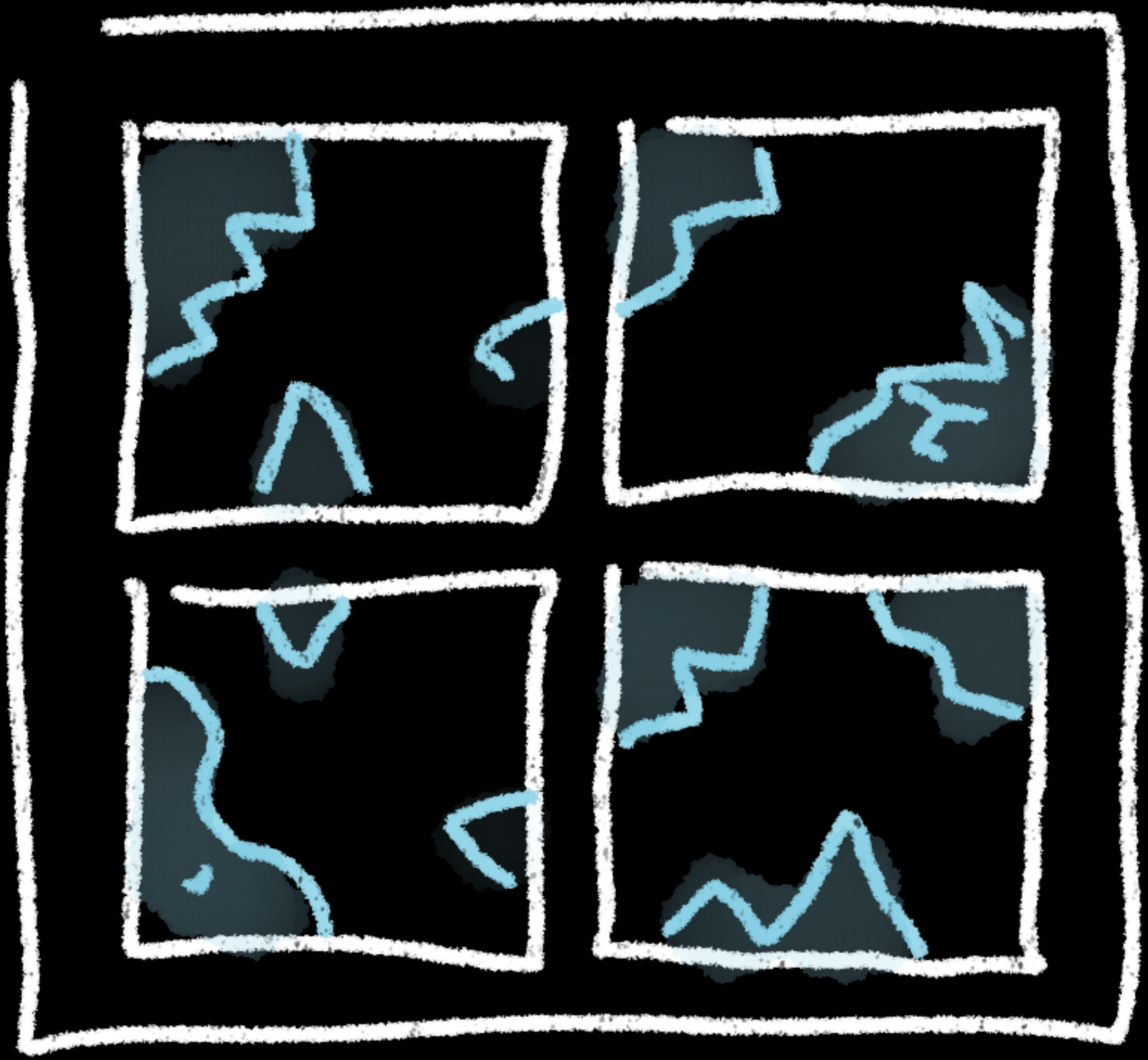
a good CI/CD indicator

“we don’t know when the build is broken” get the pipeline status into the physical spaces



“only Bob can change Jenkins”





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

judge judge
judge

modern devops

toolchains and processes reflect
cloud native apps and cultural transformation

many, single-tenant toolchains

hybrid and multi-cloud toolchains and deployments

toolchains support lean delivery processes and business agility

heritage devops

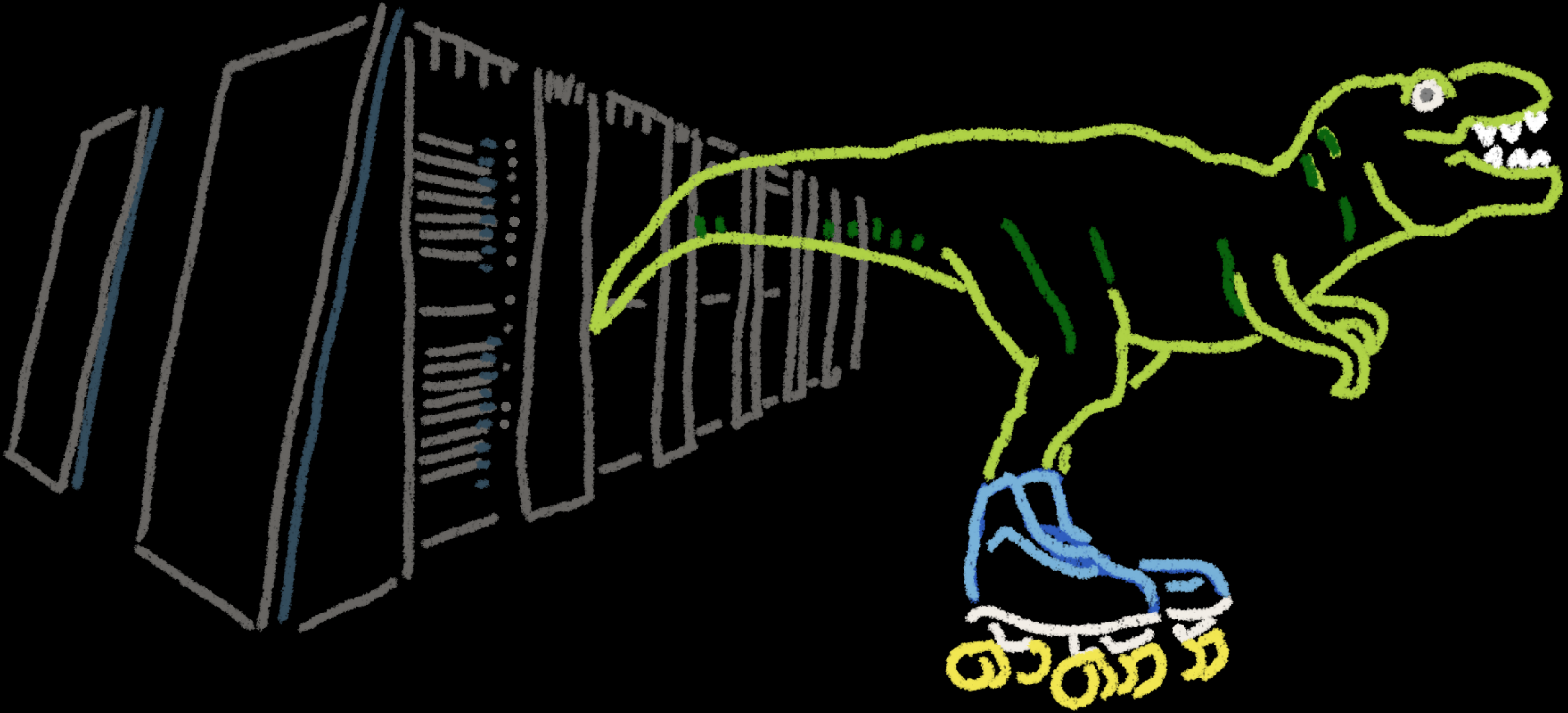
toolchains and processes reflect
heritage apps and cultural inertia

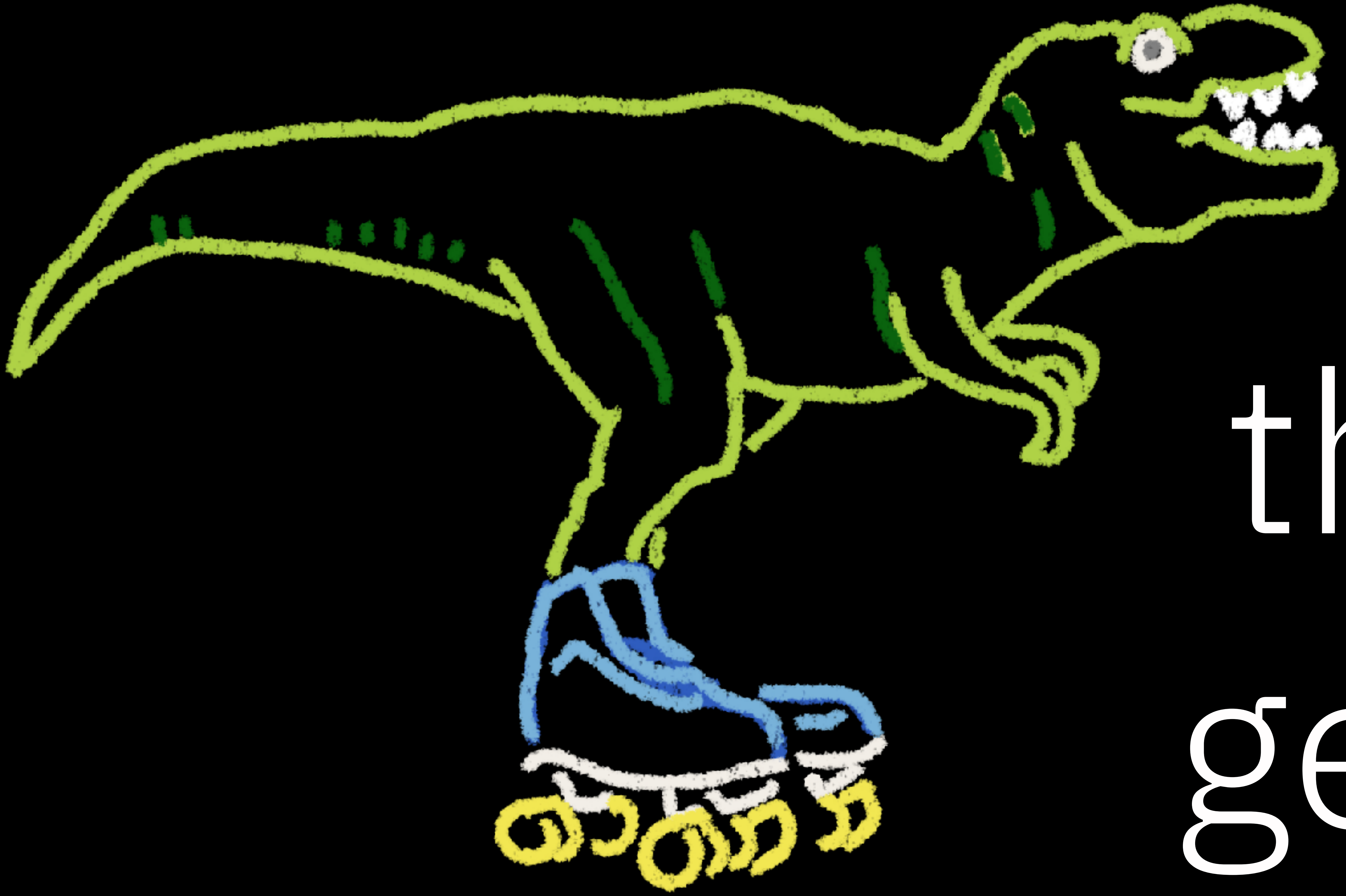
shared, multi-tenant toolchain “backbone”

on-premise automation tools

release management and dependency coordination are hard

“you’ll be coding on the mainframe”





this can get tiring

transformation
endurance

remember the **why**

@holly_cummins

IBM **Cloud** Garage