

# Cloud Native is About Culture, Not Containers

(how to not fail at cloud native)



Holly Cummins  
IBM Garage  
`@holly_cummins`



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

Please

Ask questions  
through the app

📢 *Rate Session*

Thank you!





what **is** cloud native?



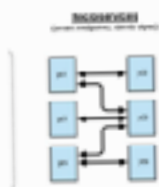
**Daniel Bryant**

@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



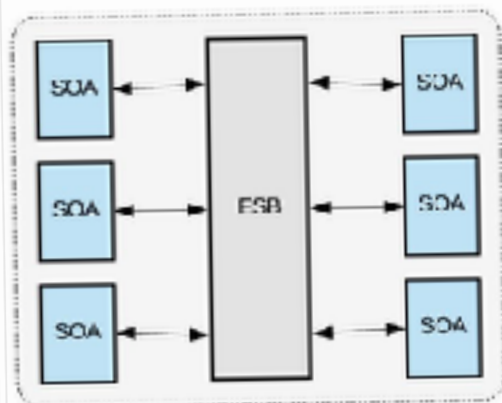
**Daniel Bryant**

@danielbryantuk

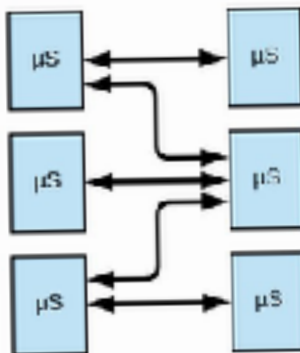
Following

**SOA/ESB**

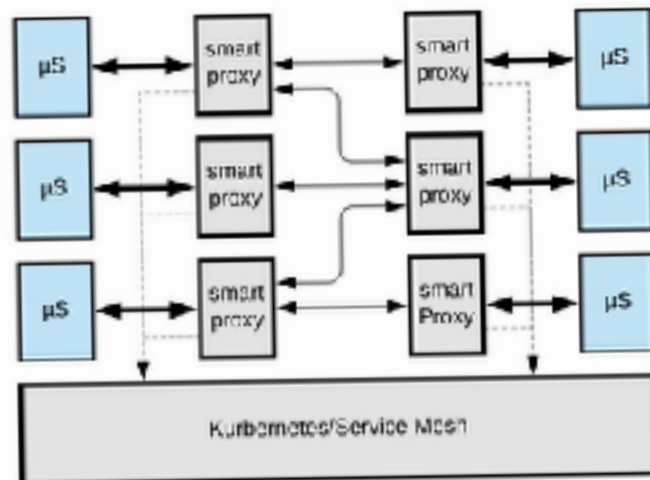
(smart pipes, dumb endpoints)

**Microservices**

(smart endpoints, dumb pipes)

**Cloud Native**

(smart platform, dumb services)



(a great  
article, btw)

www.infoq.com/articles/microservices-post-kubern...

**InfoQ**  
En | 中文 | 日本 | Fr | De  
1,201,580 Aug unique visitors

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

Software Development Conference San Francisco Nov 6-8, 2018 London Mar 4 - 5, 2019

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

Like / Posted by [Bilgin Ibrayev](#), reviewed by [Daniel Bryant](#) on Sep 01, 2018. Estimated reading time: 8 minutes 1 [Discuss](#)

Share [+](#) [i](#) [t](#) [y](#) [d](#) [f](#) [e](#)


[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 programs features that include...

### 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 

(a great  
article, btw)

The screenshot shows a web browser displaying an article on the InfoQ website. The browser's address bar shows the URL `www.infoq.com/articles/microservices-post-kubern...`. The InfoQ logo is in the top left, with navigation links for Development, Architecture & Design, AI, ML & Data Engineering, Culture & Methods, DevOps, and Videos with Transcripts. A banner for the 'Software Development Conference San Francisco Nov 6-8, 2018' is visible. The article title 'Microservices' is prominently displayed in the center. Below the title, there are social sharing icons and a 'Reading List' button. The 'Key Takeaways' section lists several points about microservices architecture and Kubernetes. On the right, there is a 'RELATED CONTENT' section with links to other articles and small profile pictures of the authors.

InfoQ  
En | 中文 | 日本 | Fr | De  
1,201,580 Aug unique visitors

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





Software Development Conference  
San Francisco Nov 6-8, 2018  
London Mar 4 - 5, 2019

Streaming Machine Learning Reactive Microservices Containers .NET All topics

You are here: InfoQ Home » Articles » Microservices

# Microservices

8 minutes | 1 Discuss


Share    


**Key Takeaways**

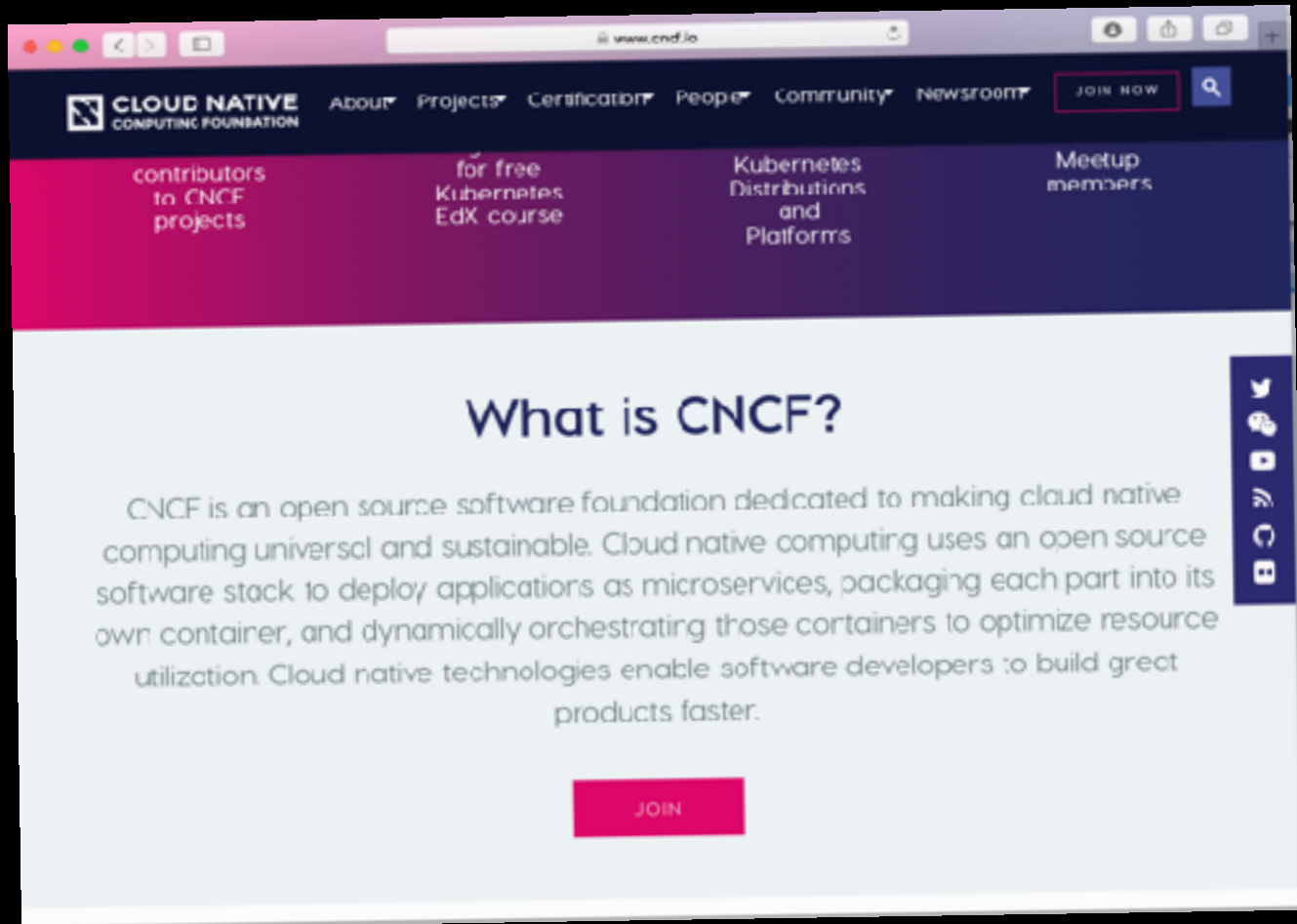
- The microservice architecture is still the most popular architectural style for distributed systems. But Kubernetes and the cloud native movement has redefined certain aspects of application design and development at scale.
- On a cloud native platform, observability of services is not enough. A more 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 concepts features that include...

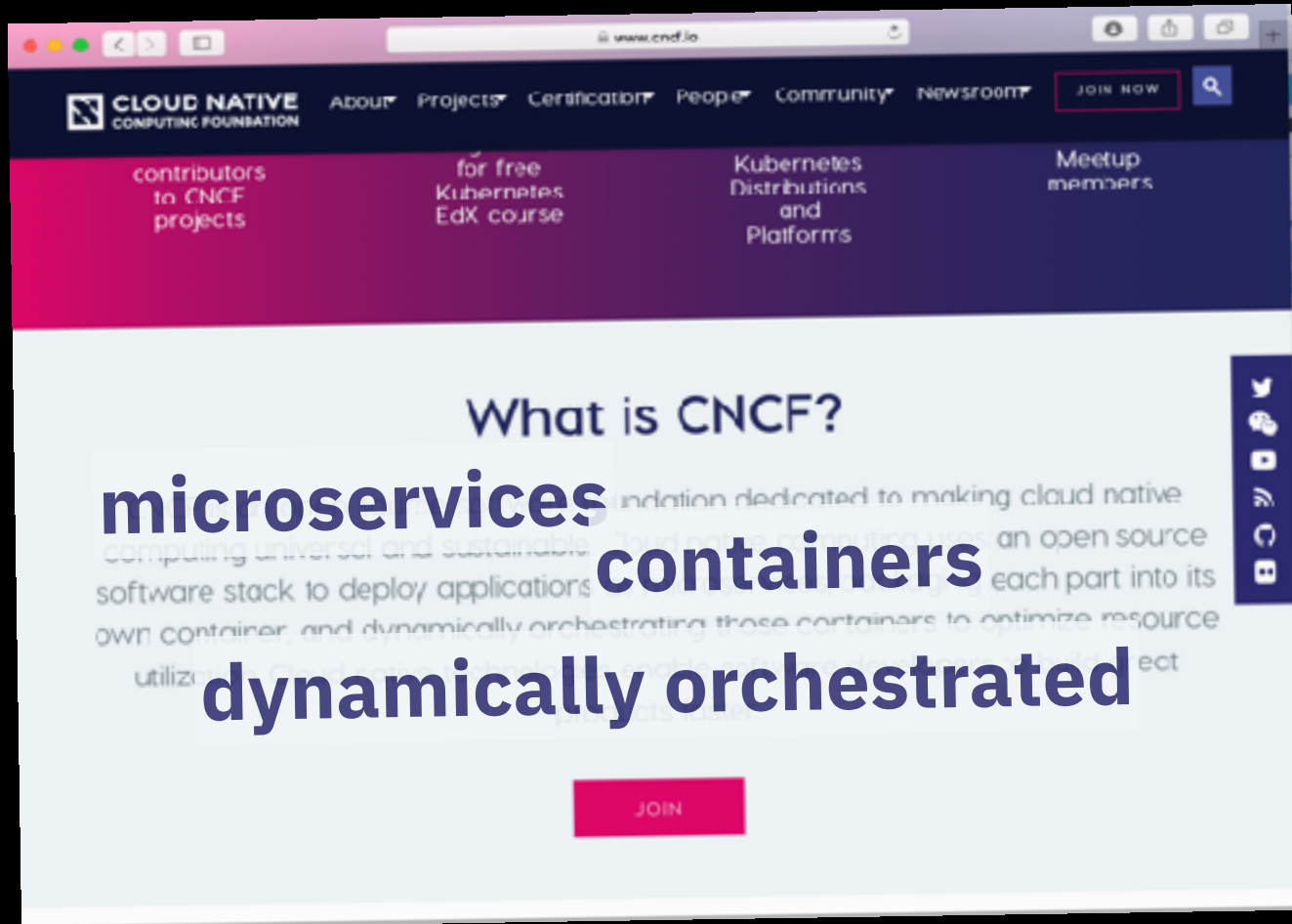
**RELATED CONTENT**

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 





**goto;**

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

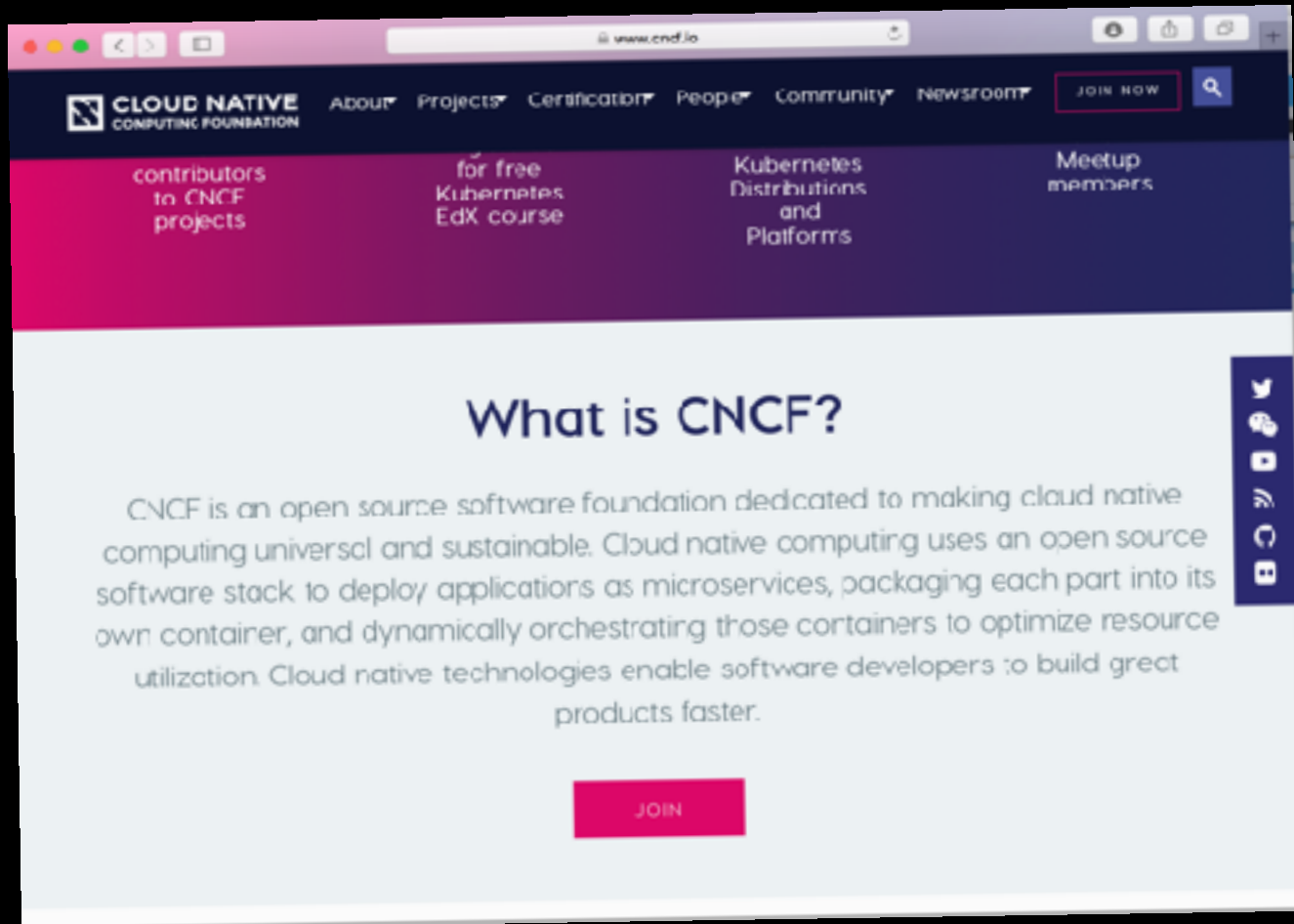


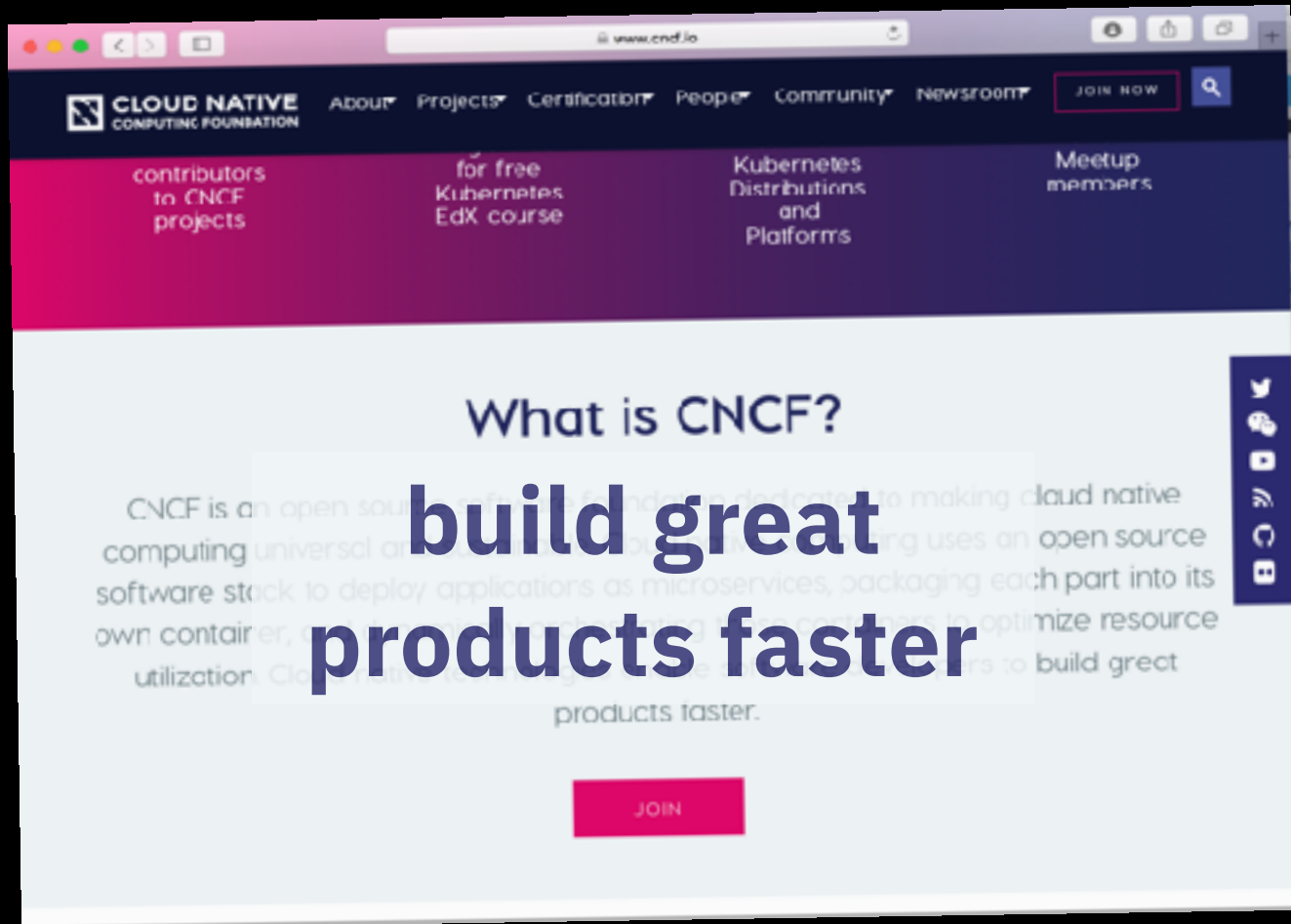
**goto;**

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



Holly





# why?

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



# CV-driven development.



“my CV looks dull”  
is not a good reason  
to go cloud native

# why cloud?

cost





e l a s t i c i t y



# exotic capabilities



why cloud native?



12  
factors

# 12 factors

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





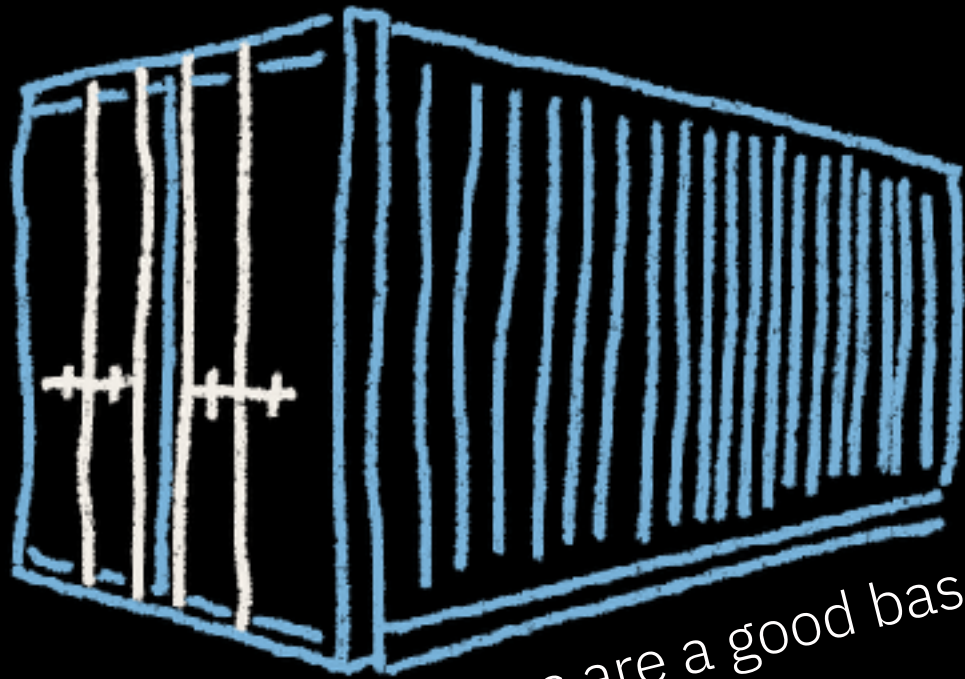
cloud native is not a  
synonym for  
‘microservices’



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

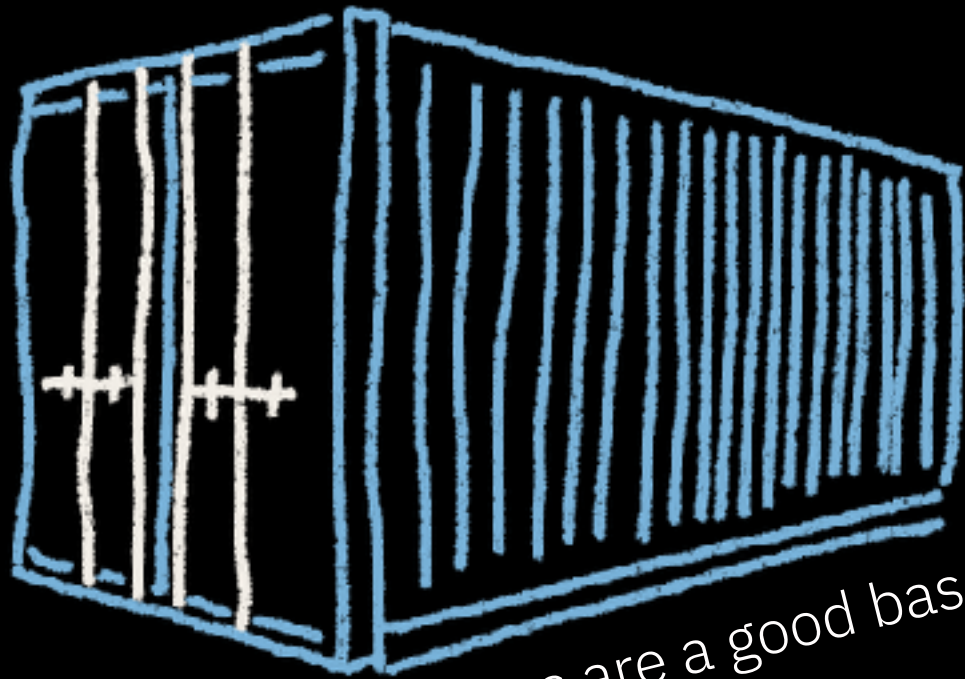
if 'cloud native' has to be a  
synonym for anything, it would be  
'idempotent'

which definitely needs a synonym



containers are a good base

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



containers are a good base

you do not need intra-app http  
communication to be cloud native



complexity adds expense

unnecessary complexity  
adds unnecessary expense



space pencil

space pencil

\$128.89

space pencil

space pen

\$128.89

space pencil

\$128.89

space pen

\$2.39

space pencil

\$128.89

+medical bills

space pen

\$2.39

accidental  
complexity

essential  
complexity

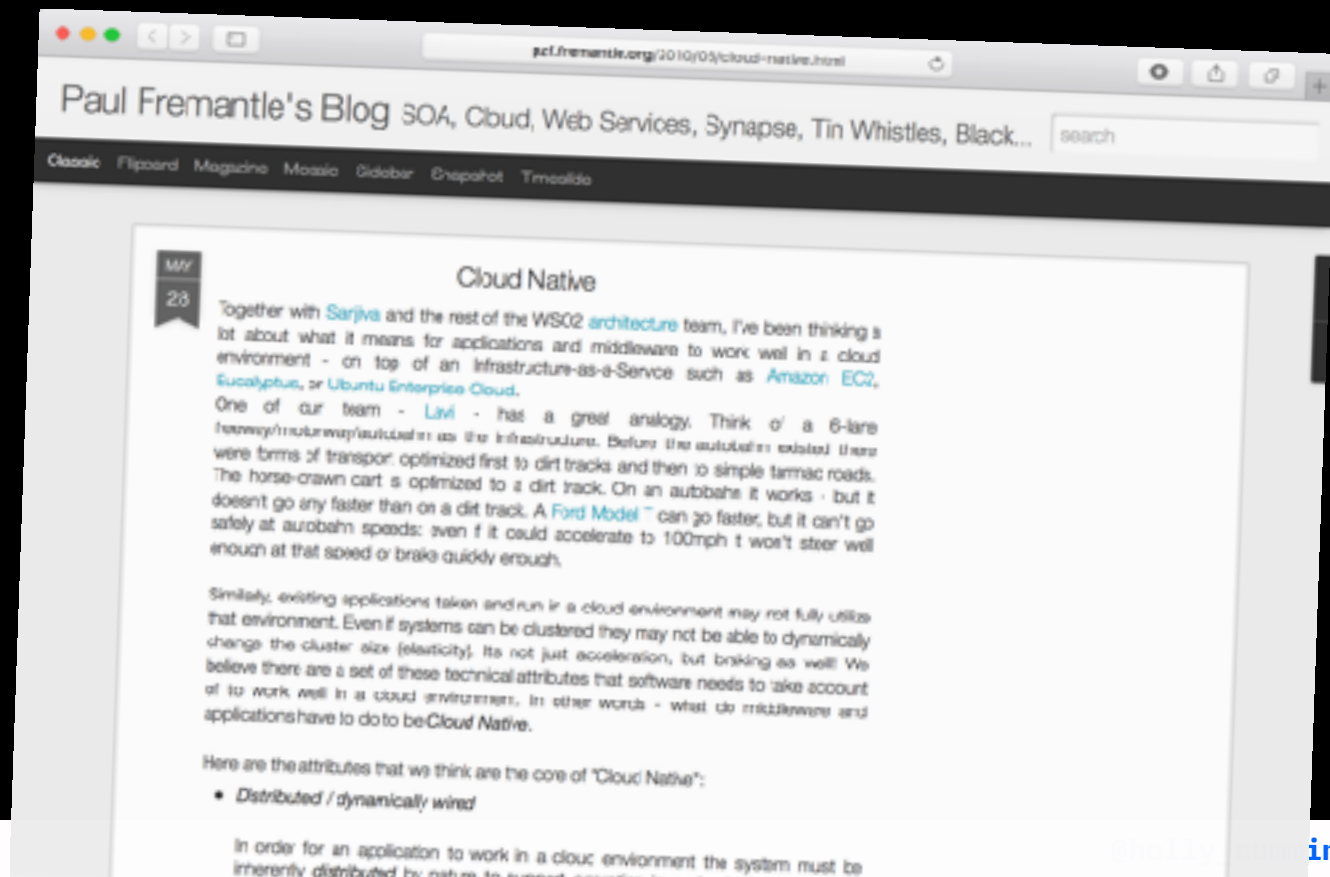
accidental  
complexity

essential  
complexity



“this will impress my  
boss” complexity

# 2010 the dawn of cloud native





behaves well on the cloud

behaves well on the cloud  
written for the cloud

this is all how we **run** our  
application, not what's **in** it

speed

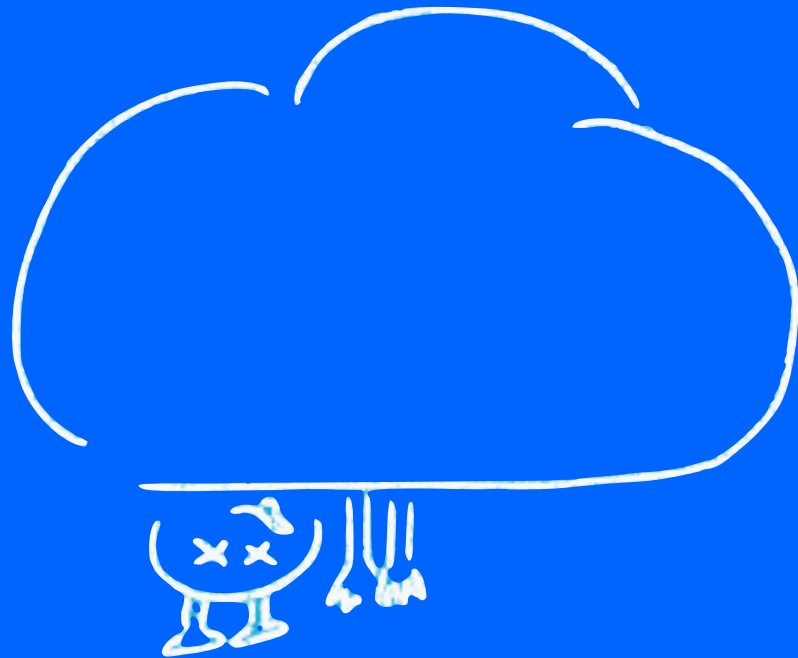
speed

what's the point of getting the  
same old **stuff** to market faster?

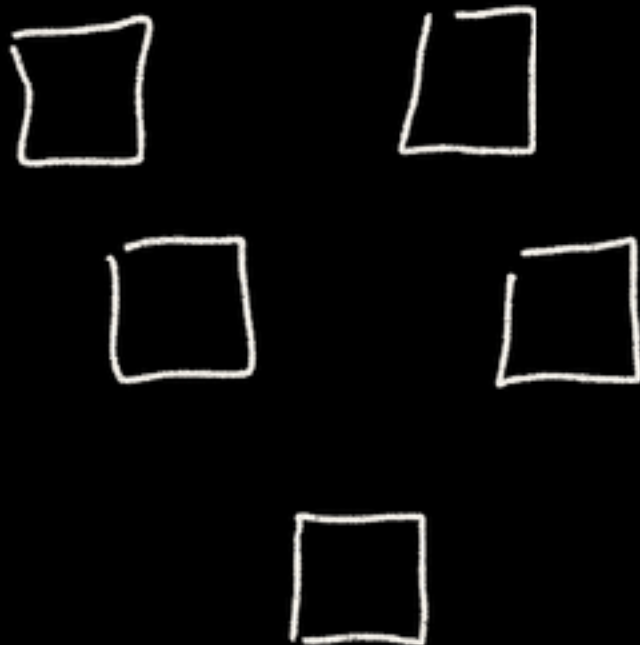
what's the point of being able to  
respond to the market, if you **don't**?

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

# how to fail at cloud native

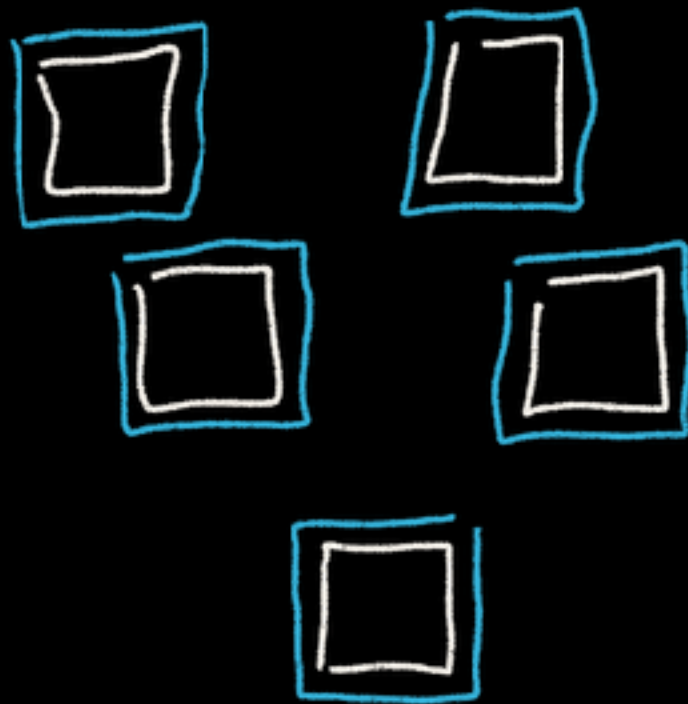


“each of our microservices has duplicated the same object model ... with twenty classes and seventy fields”



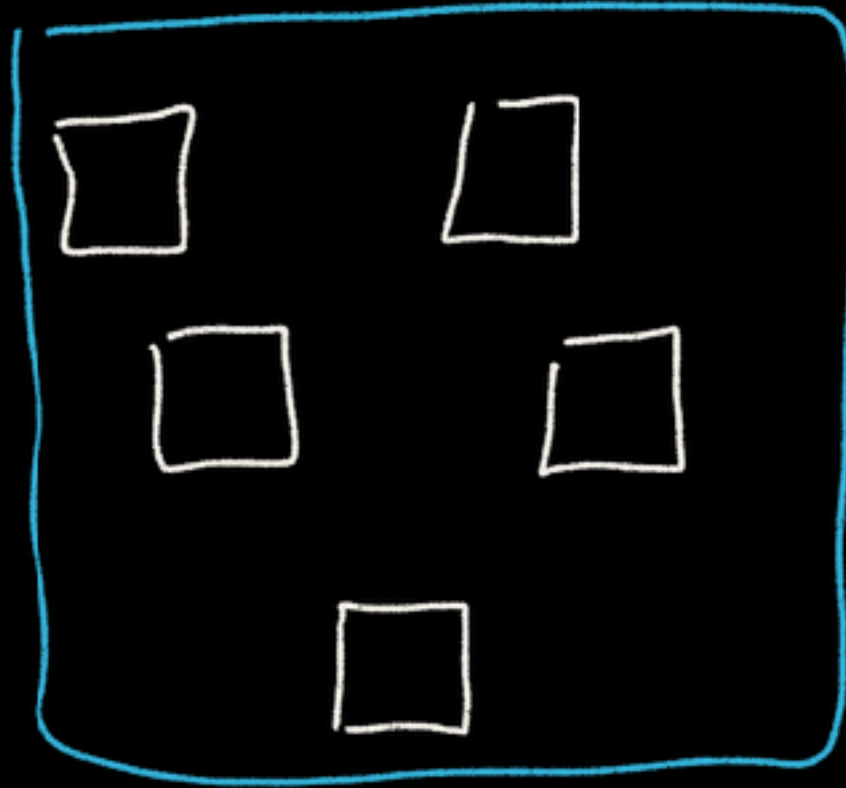
Microservice

Domain



Microservice

Domain



“every time we change  
code, something breaks”

# distributed monolith

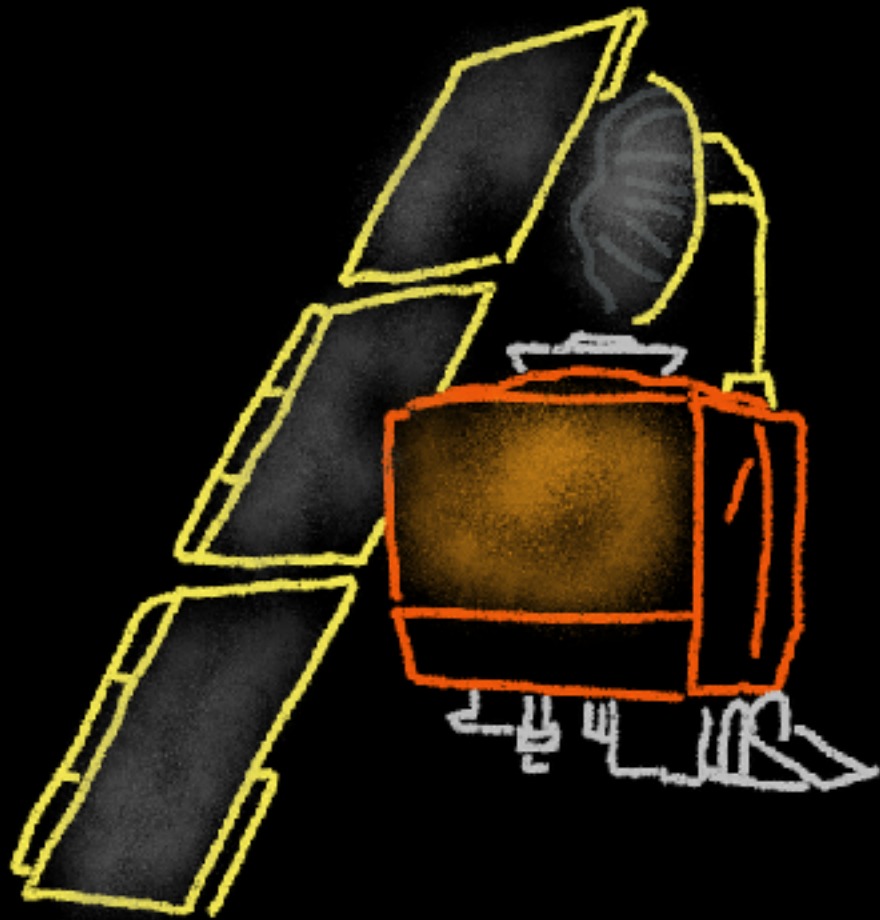


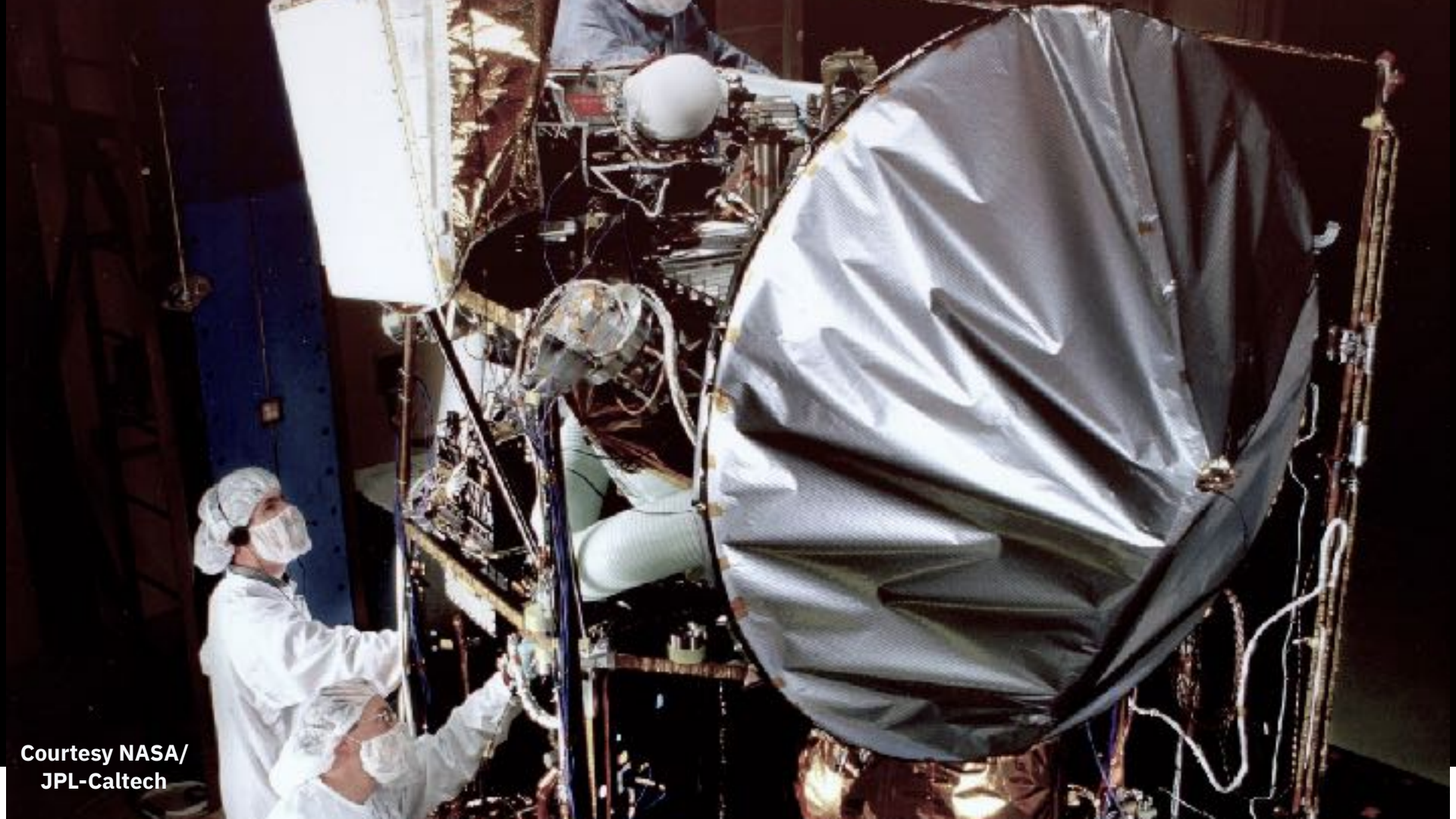
cloud-native spaghetti is still spaghetti

*(Image: Cloudy with a Chance of Meatballs.)*

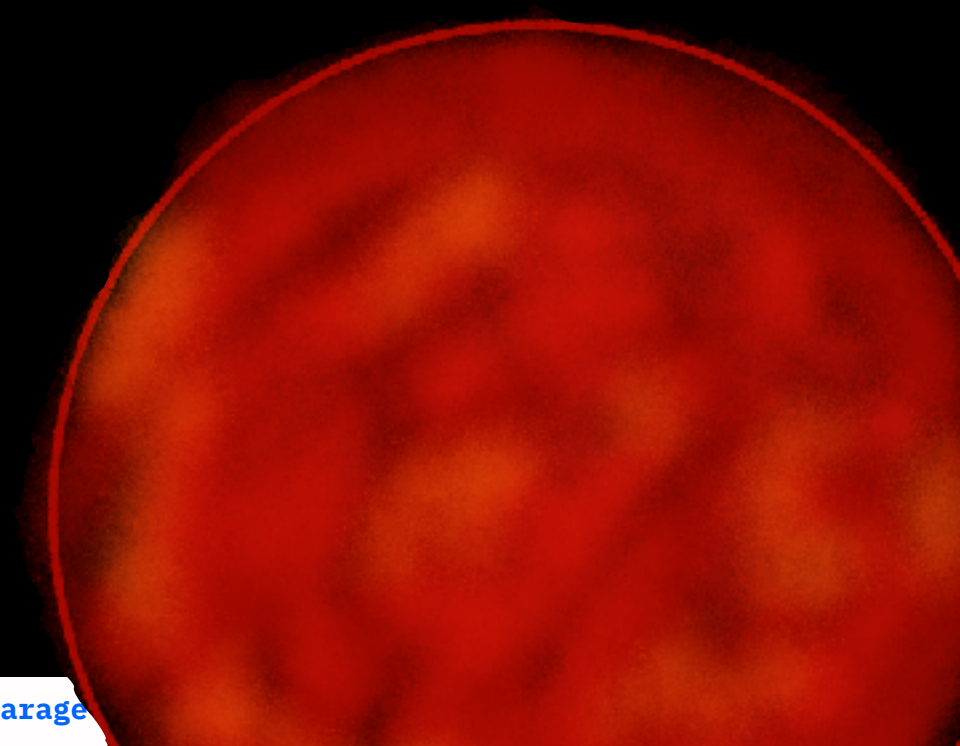
just because a system runs across 6  
containers doesn't mean it's decoupled

distributed  $\neq$  decoupled

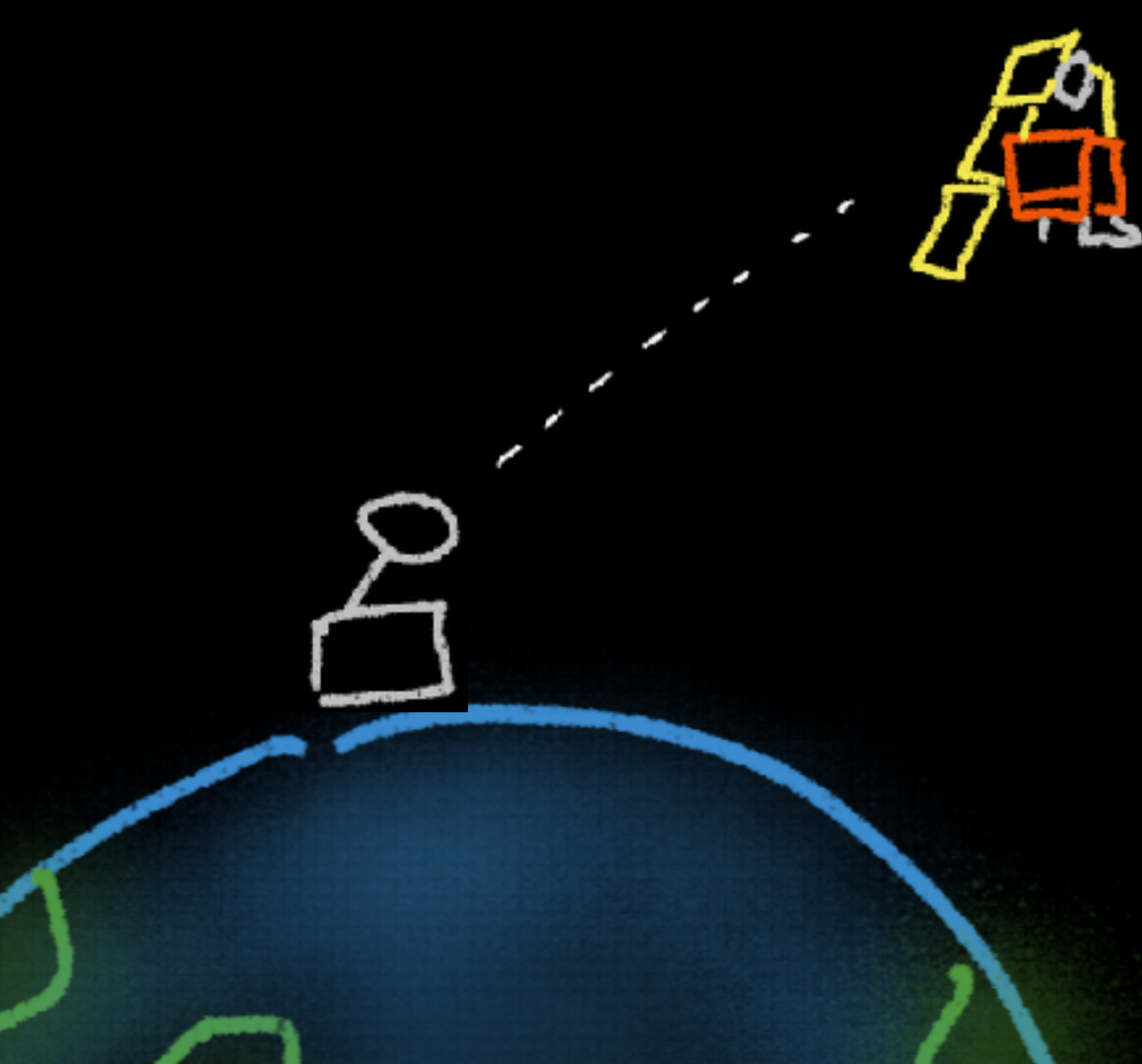


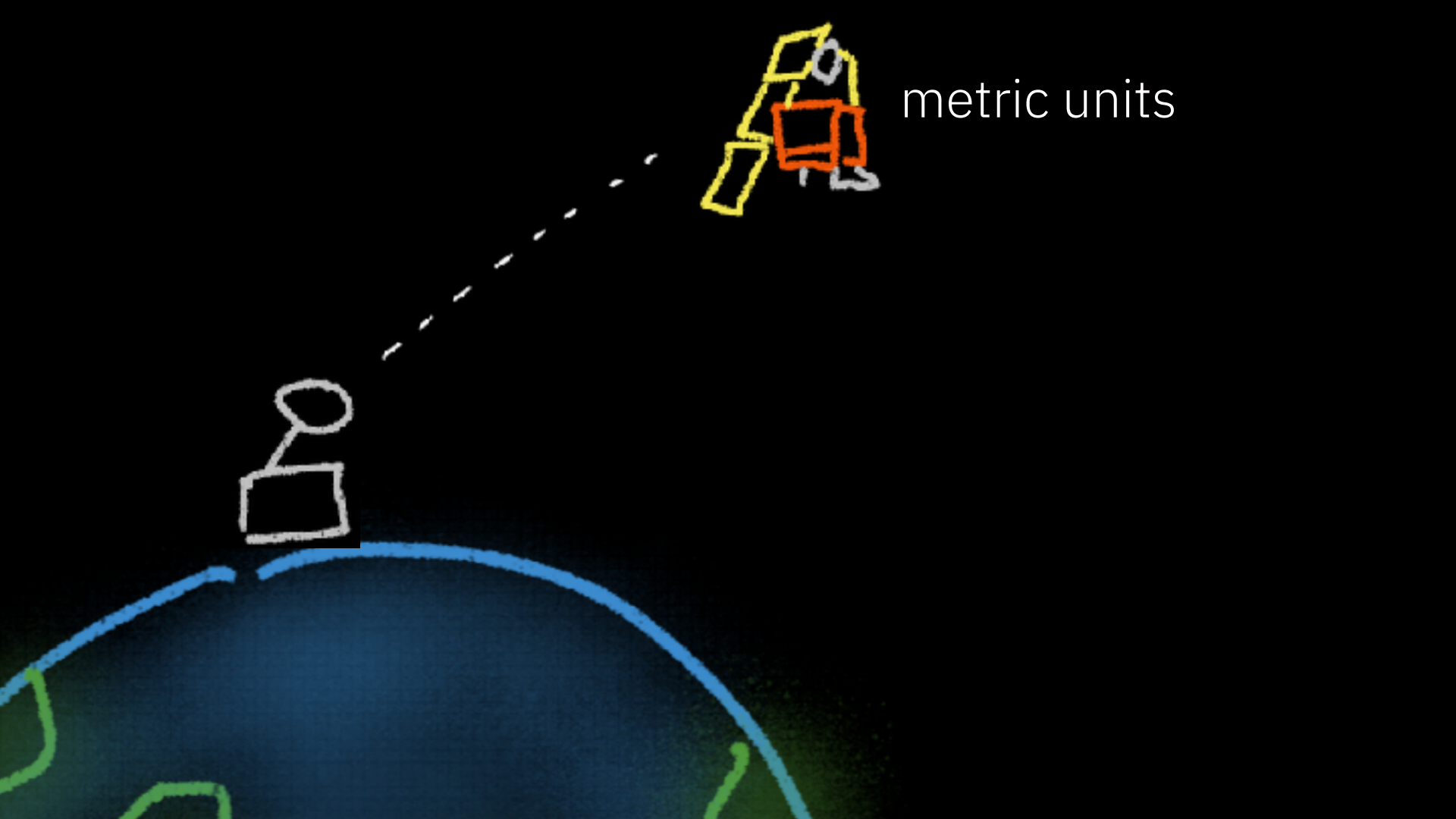


Courtesy NASA/  
JPL-Caltech









metric units

imperial  
units



metric units

imperial  
units



metric units

distributing  
did not help

microservices **need**  
consumer-driven contract tests

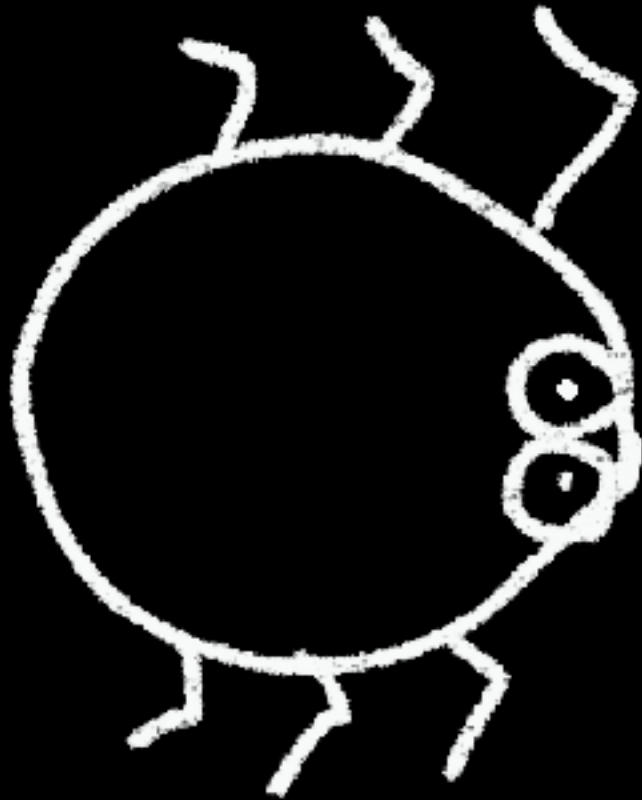
“our tests aren’t  
automated”

“we don’t know if  
our code works”

systems **will** behave in  
unexpected ways

documentation can be  
wrong

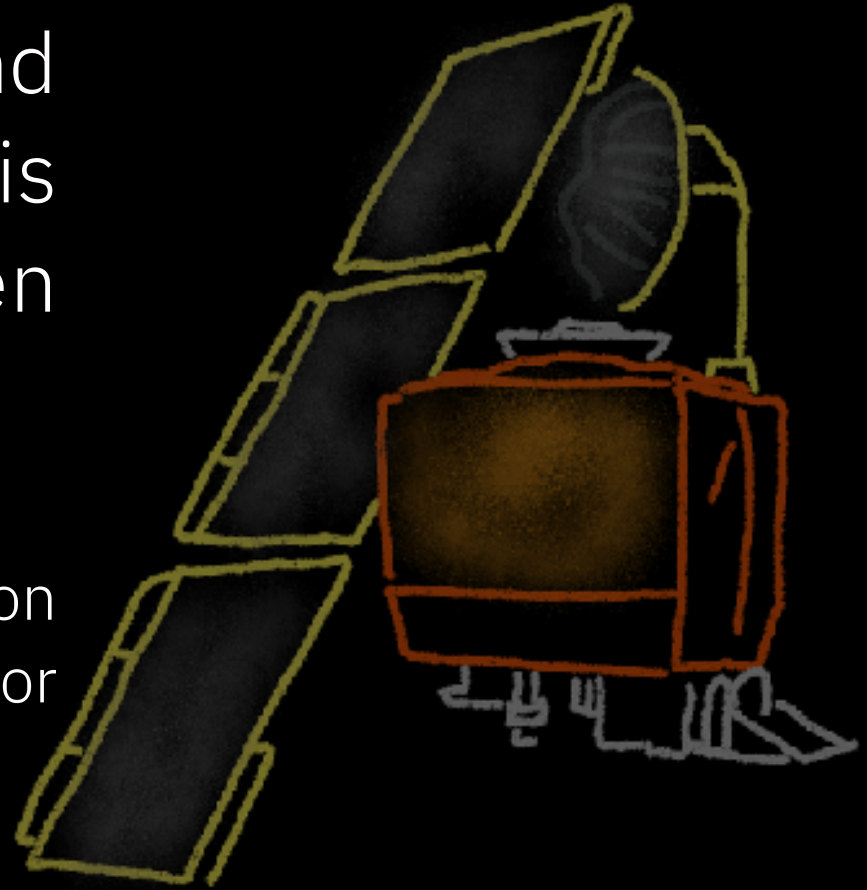
dependency updates  
can change behaviour





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

Arthur Stephenson  
Chief Investigator



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



microservices **need**  
automated integration tests

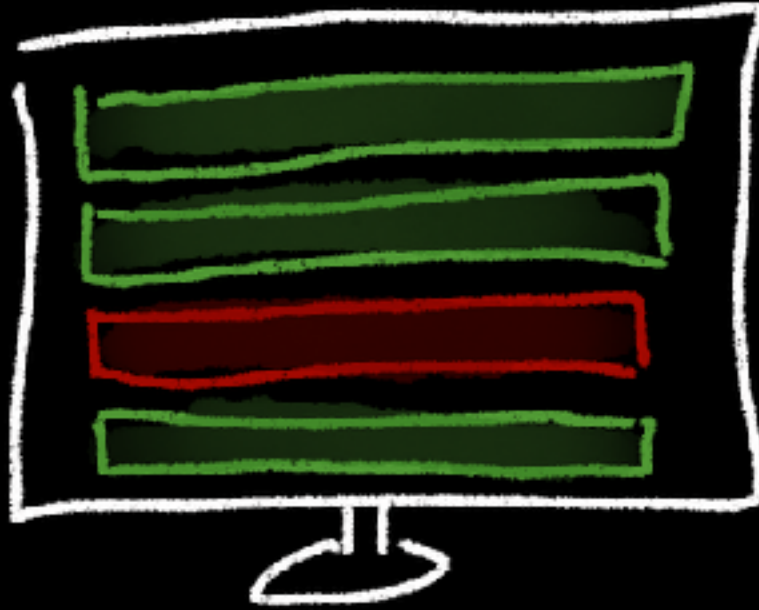


**not** a good CI/CD indicator

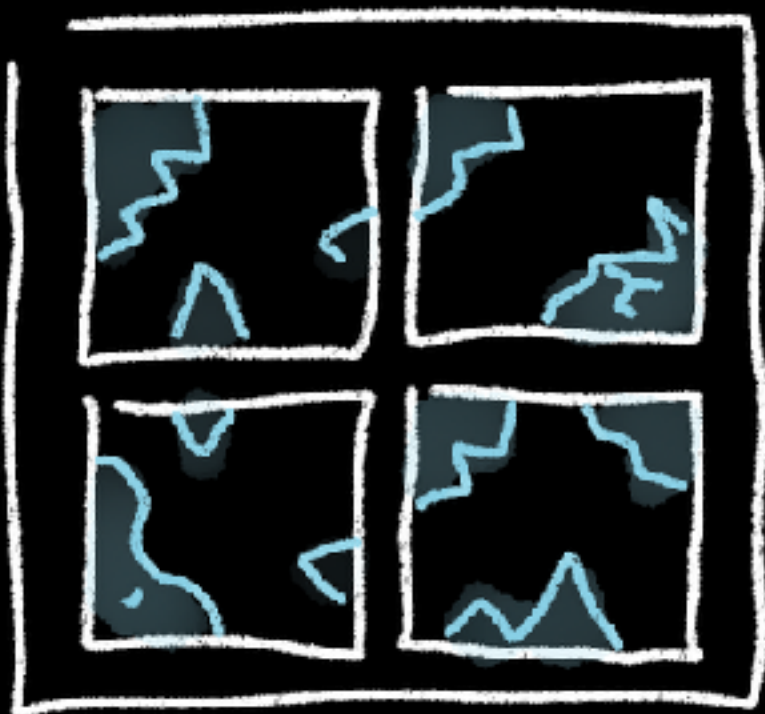


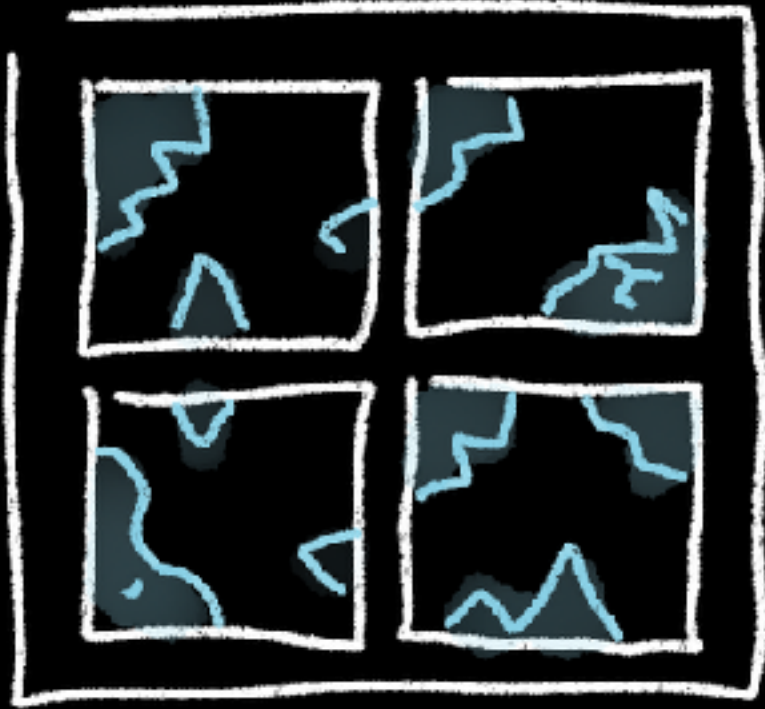
a good CI/CD indicator

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



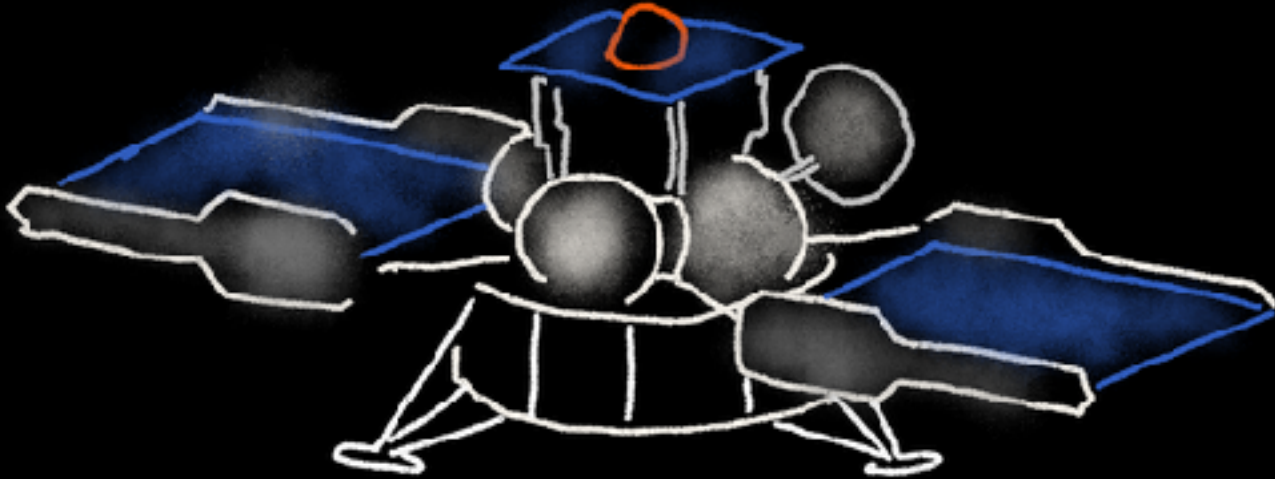
a good build radiator



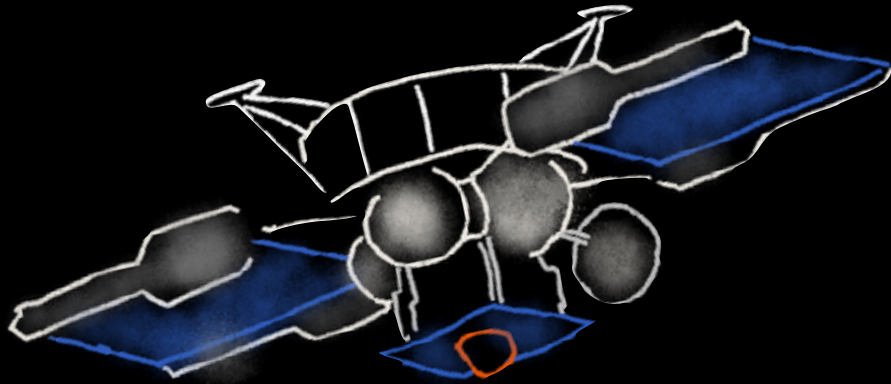


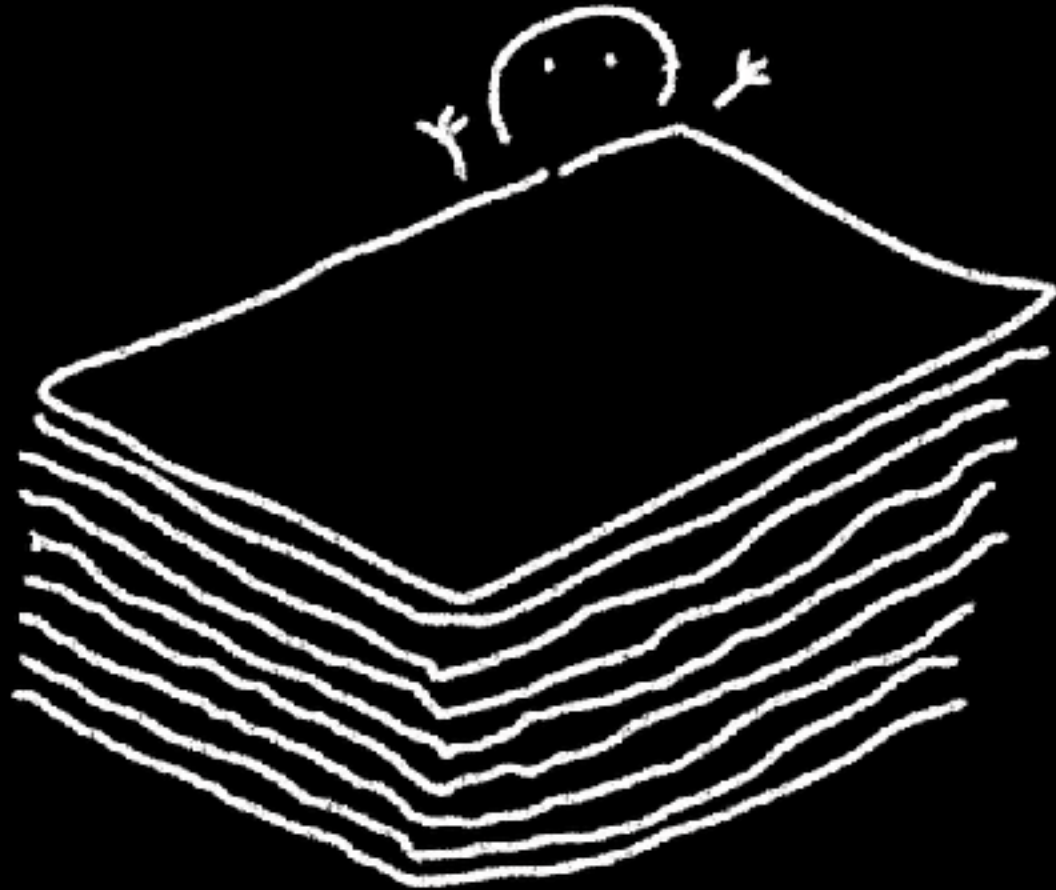
“oh yes, that  
build has been  
broken for a  
few weeks...”

# how to brick a spaceprobe

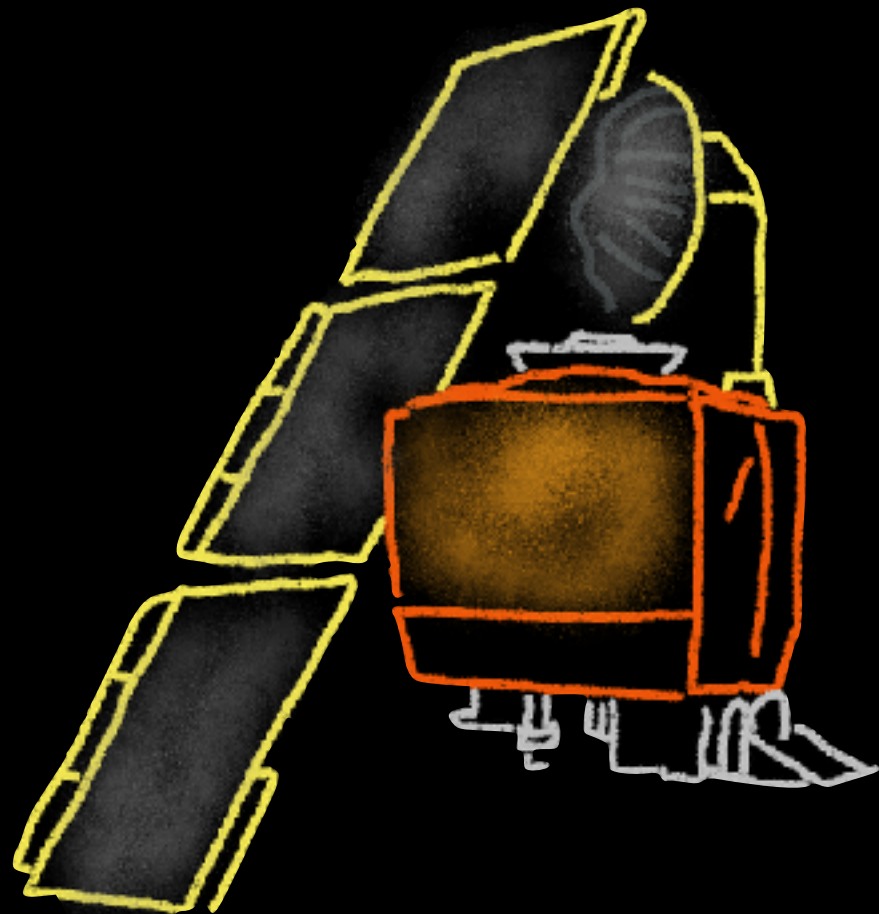


“we couldn’t get the  
automated checks to work,  
so we bypassed them”

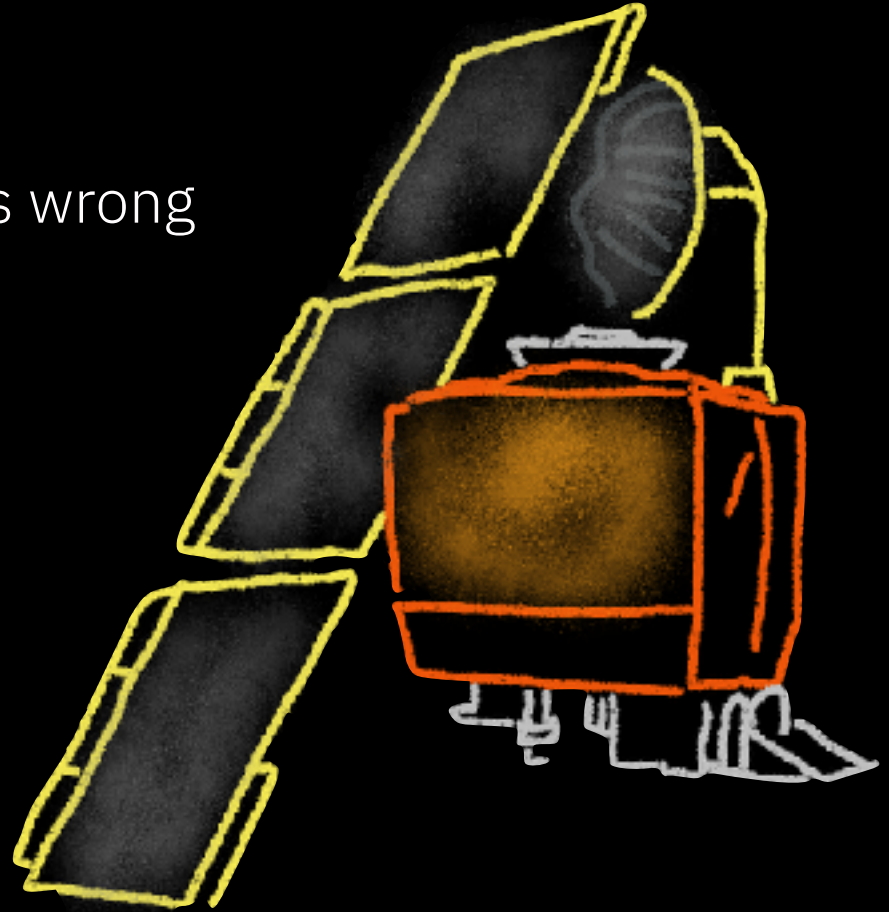




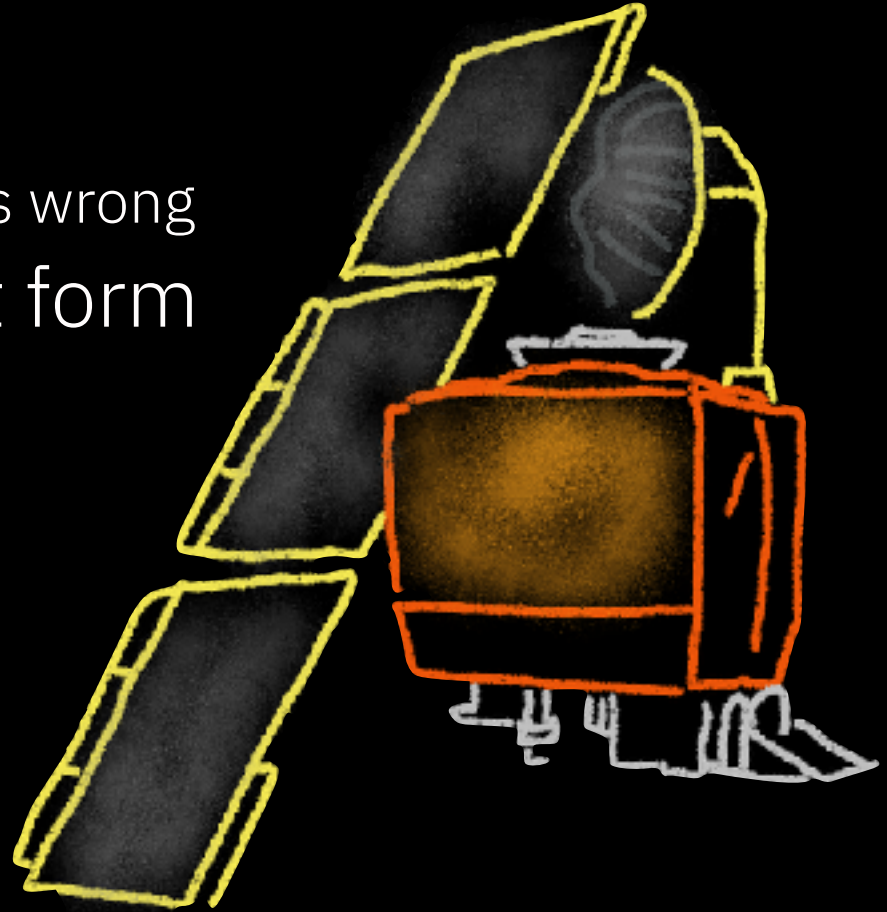
“we’ve  
scheduled the  
architecture  
board review for  
a month after the  
project ships”



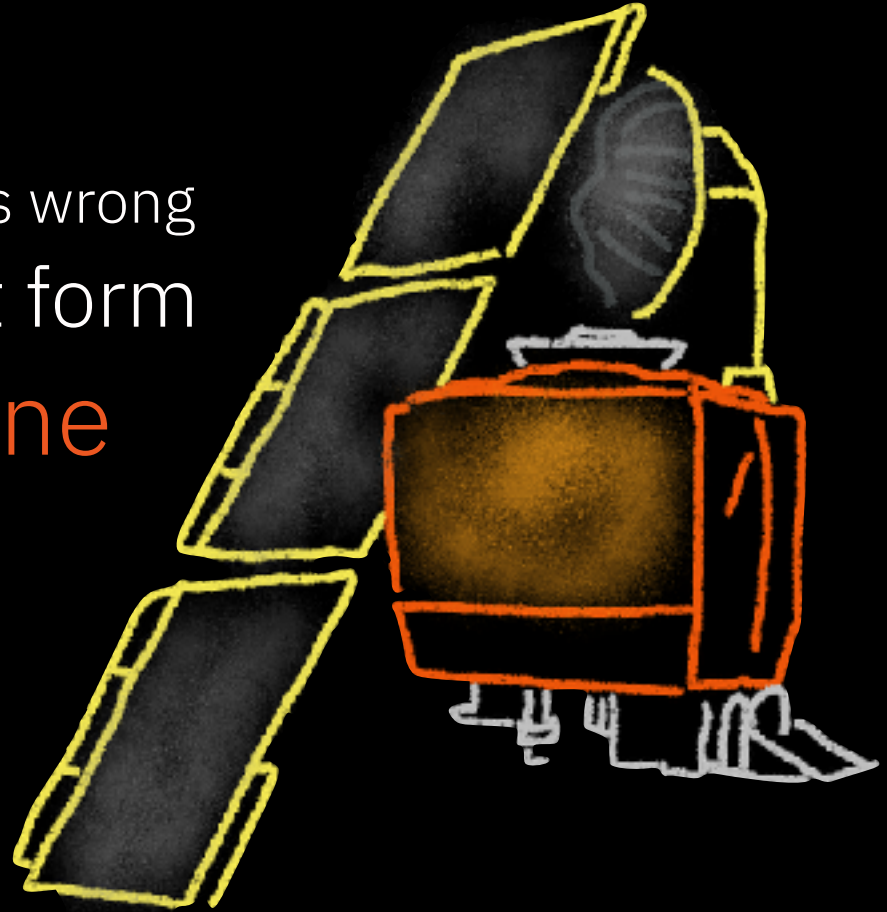
navigators warned something was wrong



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



does the process add  
value?

extreme programming is  
the right kind of rigour

extreme programming is  
the right kind of rigour

test-driven development

extreme programming is  
the right kind of rigour

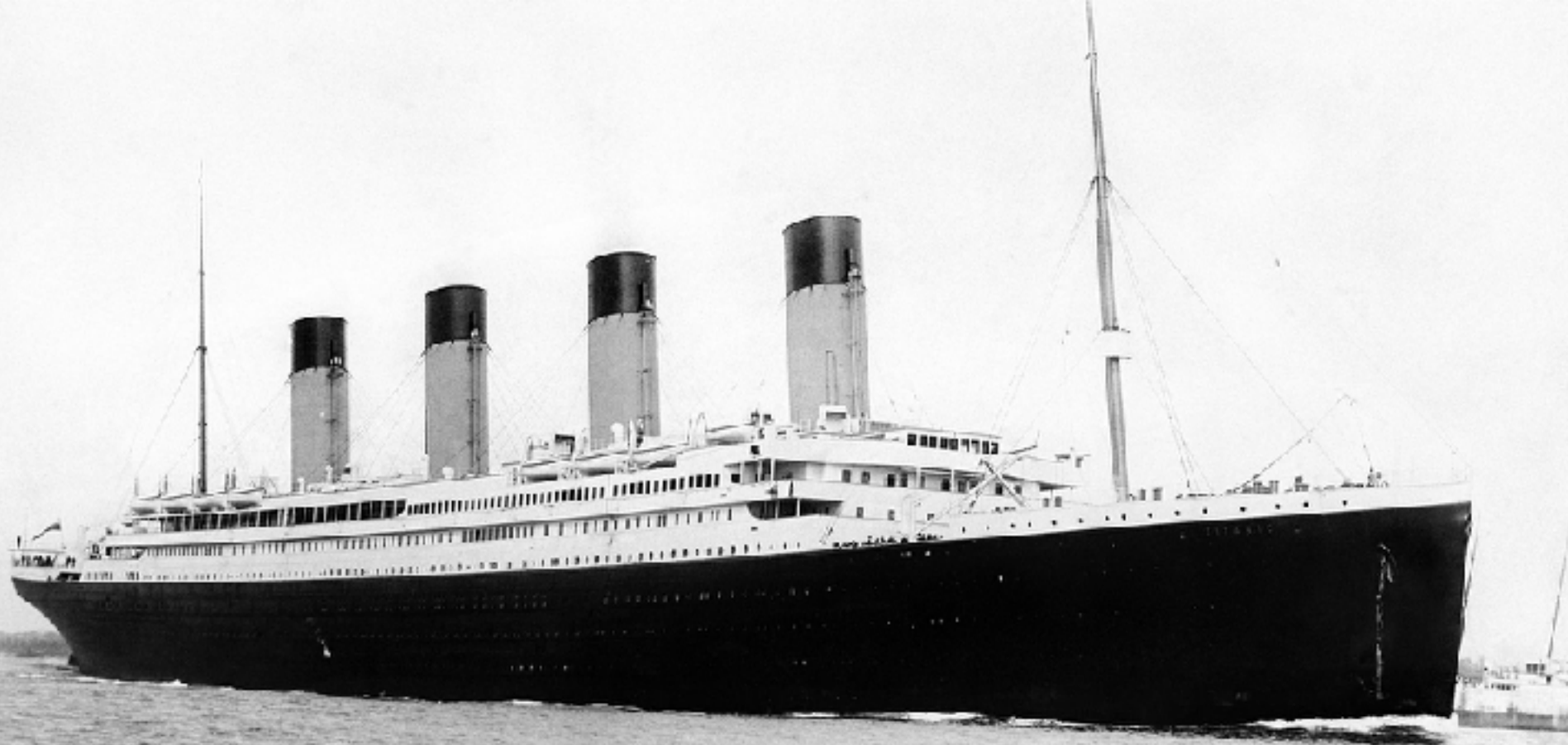
test-driven development  
pair programming

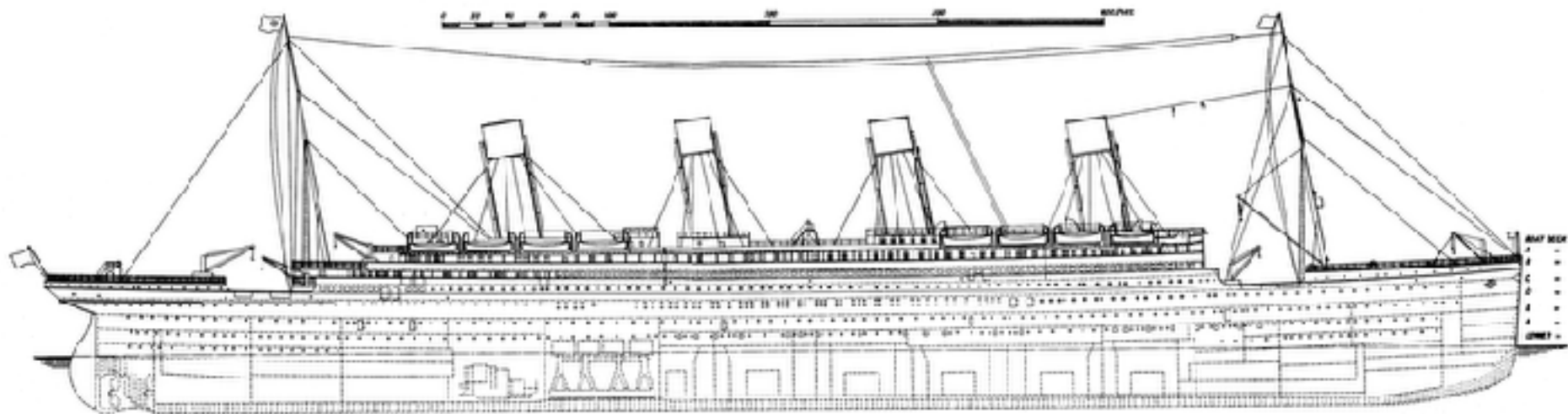
# extreme programming is the right kind of rigour

test-driven development  
pair programming  
optimise for feedback

“but it’s in the plan”

“but it’s not in the plan”





lots of bulkheads

it was too big

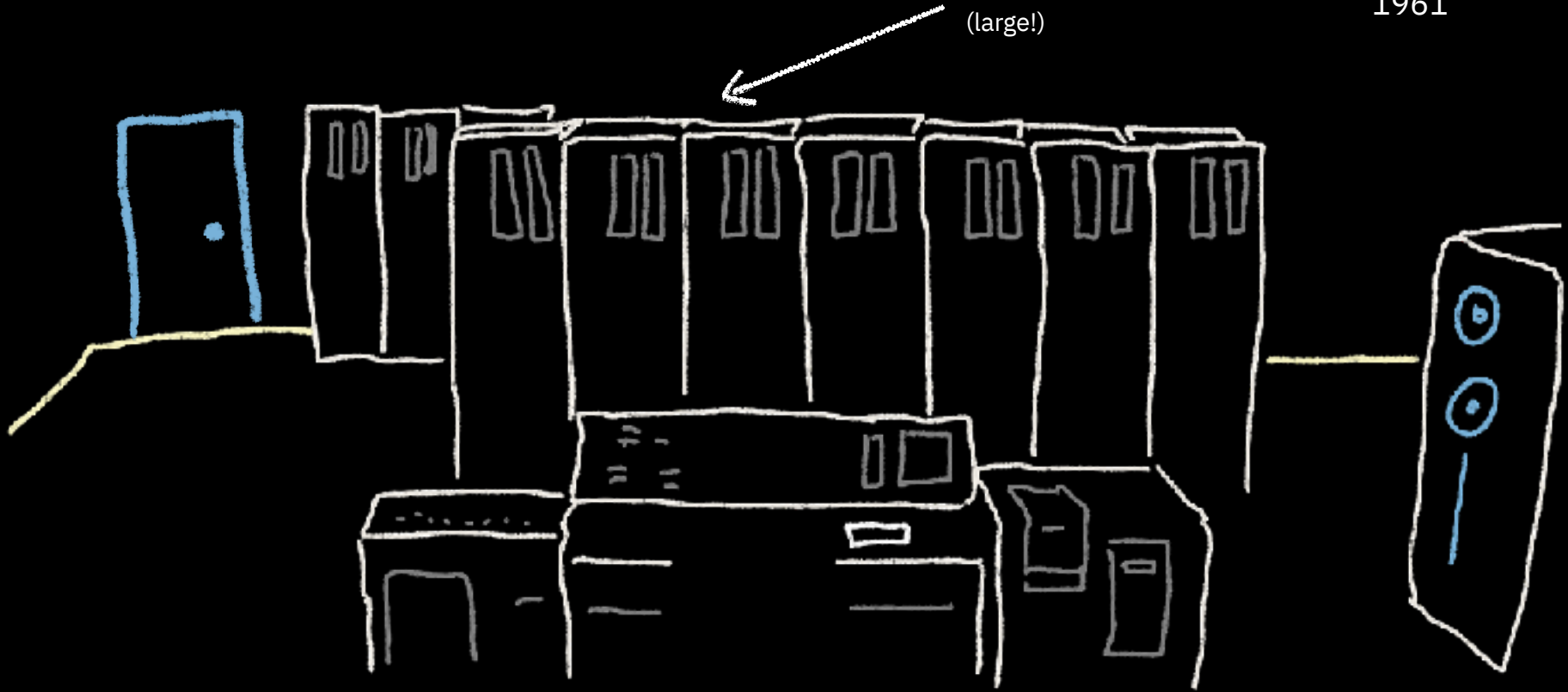
lookouts **saw** the iceberg  
but the ship wasn't nimble enough to avoid it

NASA Langley  
1961



NASA Langley  
1961

IBM 7090  
(large!)



door  
(normal-sized)

IBM 7090  
(large!)

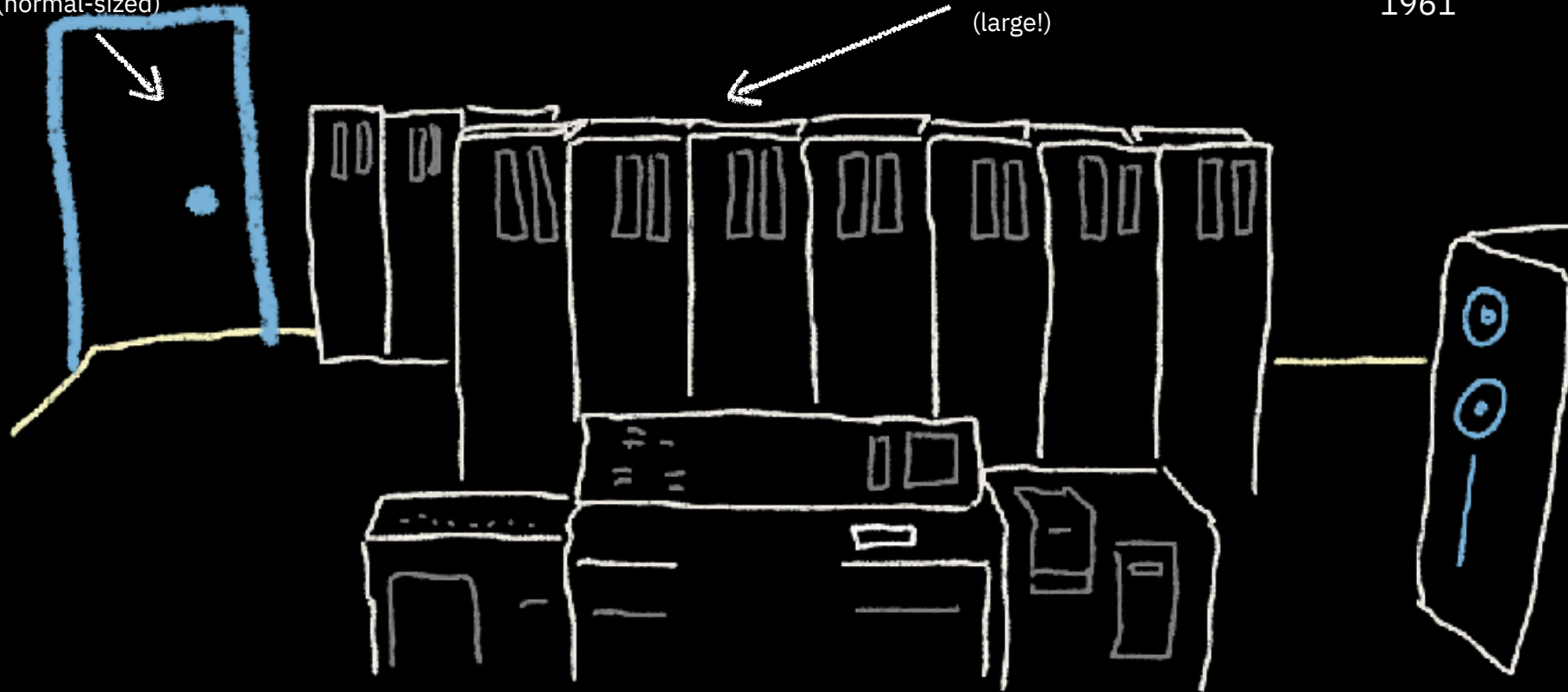
NASA Langley  
1961



door  
(normal-sized)

IBM 7090  
(large!)

NASA Langley  
1961

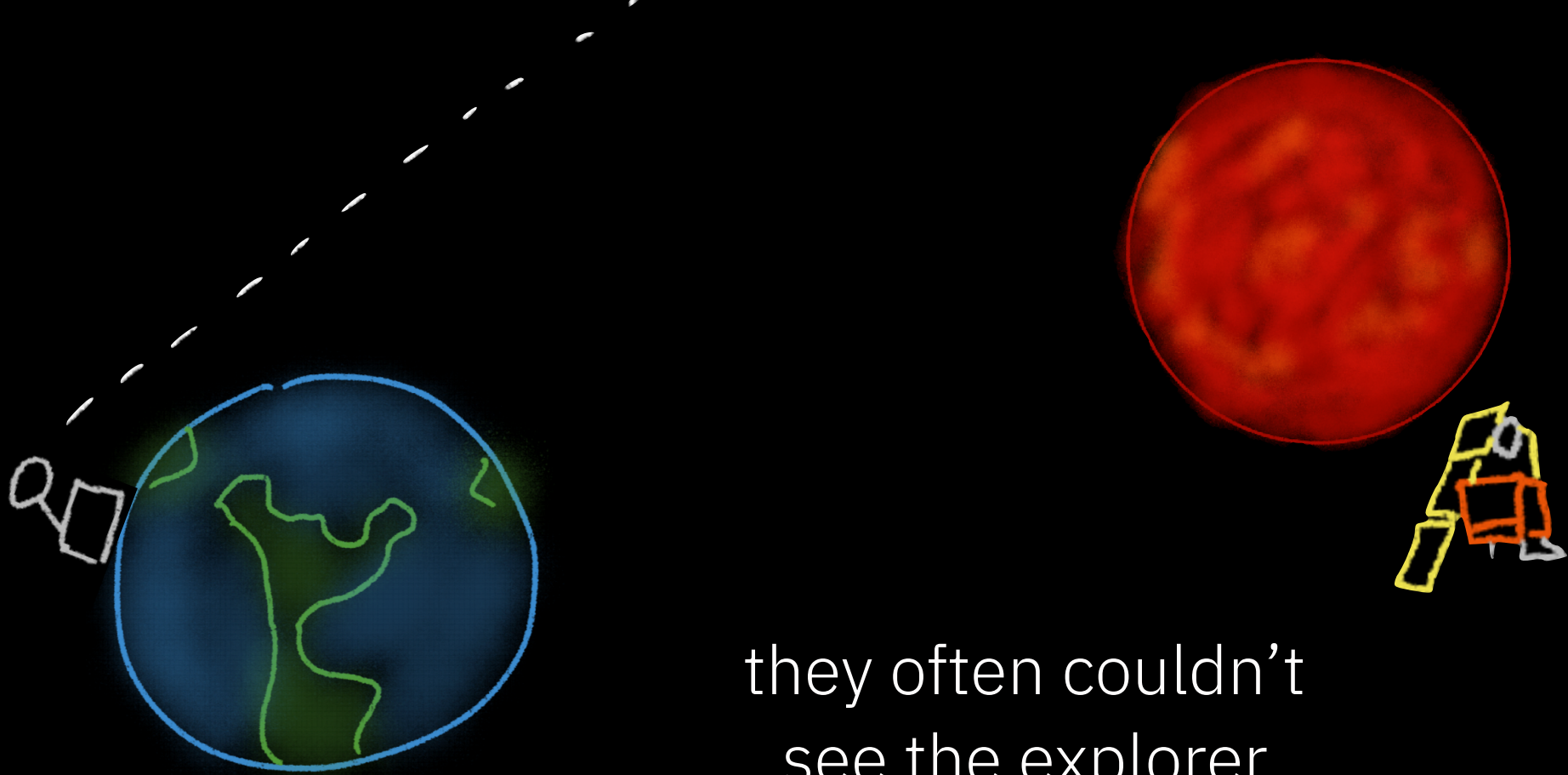


plans are **always** wrong  
success is in how you respond

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



how **not** to  
drive a car



they often couldn't  
see the explorer

feedback is good business

feedback is good engineering

an mvp hurts

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

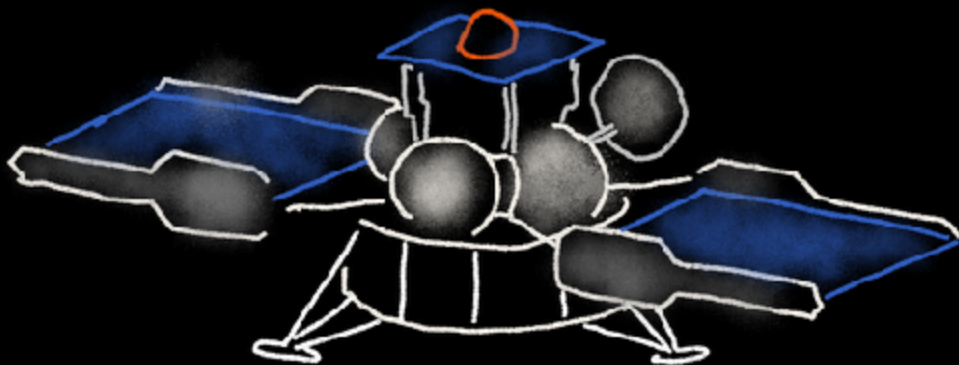
experiments can mean failure

experiments mean failure

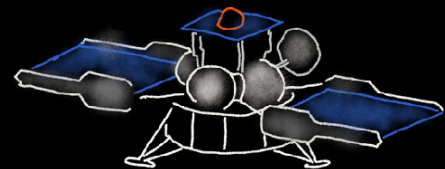


users will have  
**weird** behaviours

# optimise for recovery



# unrecoverable



bricked

manual  
intervention

handoffs

fast, but  
data lost

back in ms  
no data loss



# business recoverability

remember, users will  
have **weird** behaviours



speed



slow is  
demoralising  
for teams

fast is  
good  
business



A late change in requirements  
is a competitive advantage.

*—Mary Poppendieck*



more feedback → more accuracy



cloud rescued developers  
from tedium and toil

cloud native  
should feel  
**fun**



Please

**Remember to  
rate this session**

Thank you!





@holly\_cummins