

# cloud chaos and microservices mayhem

Holly Cummins  
IBM **Garage**  
`@holly_cummins`













**things you  
need to do  
well in 2021:**



handwashing

**things you  
need to do  
well in 2021:**



handwashing  
apps

**things you  
need to do  
well in 2021:**



**things you  
need to do  
well in 2021:**

handwashing  
apps  
ops

**things you  
need to do  
well in 2021:**

handwashing  
apps  
ops  
devops



**things you  
need to do  
well in 2021:**

handwashing

apps

ops

devops

devsecops

**things you  
need to do  
well in 2021:**

handwashing

apps

ops

devops

devsecops

**finops**










a few  
things that are  
different in the





things that are  
different in the



# management

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

That doesn't mean  
the hardware is free.

That doesn't mean the hardware  
is free.



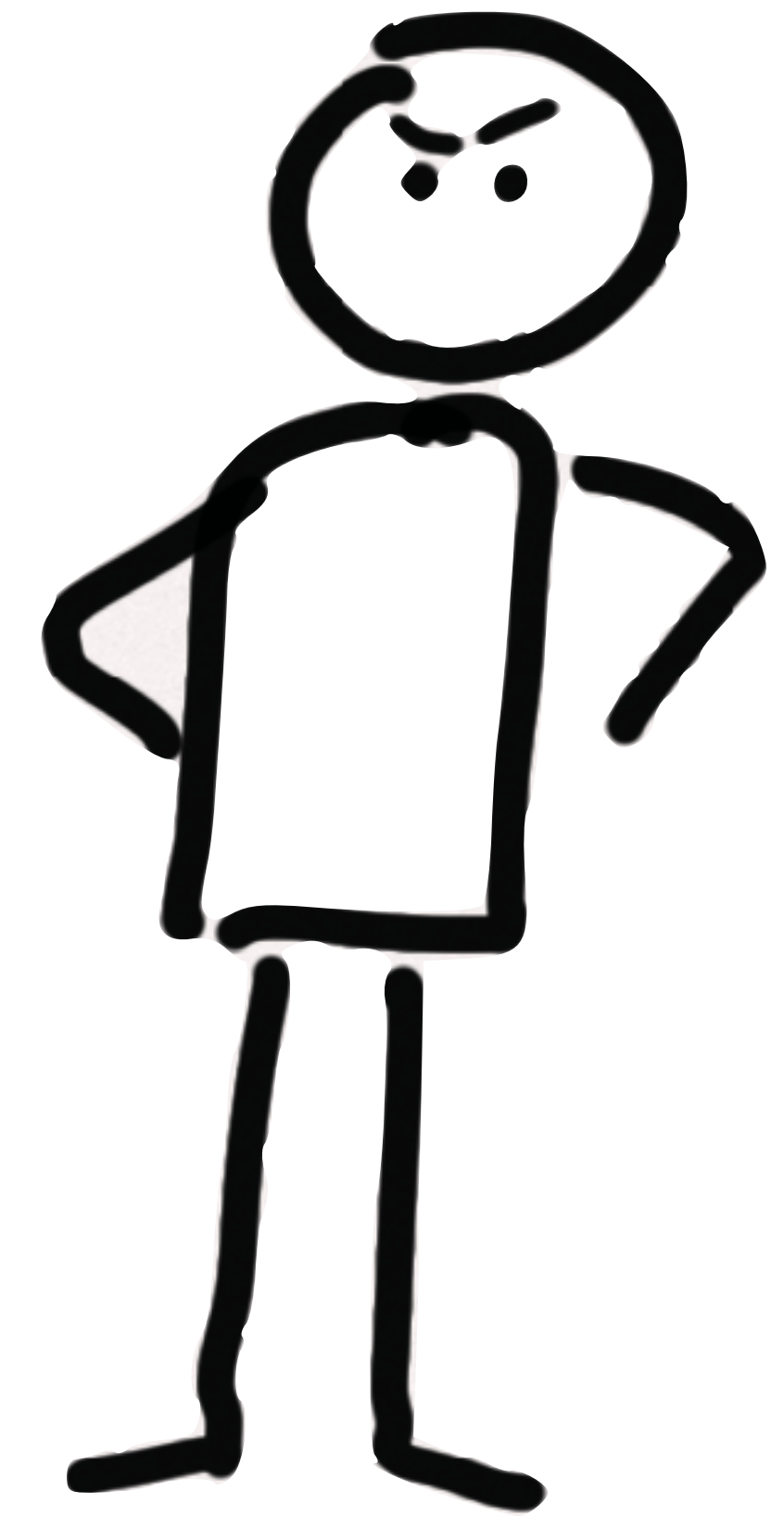
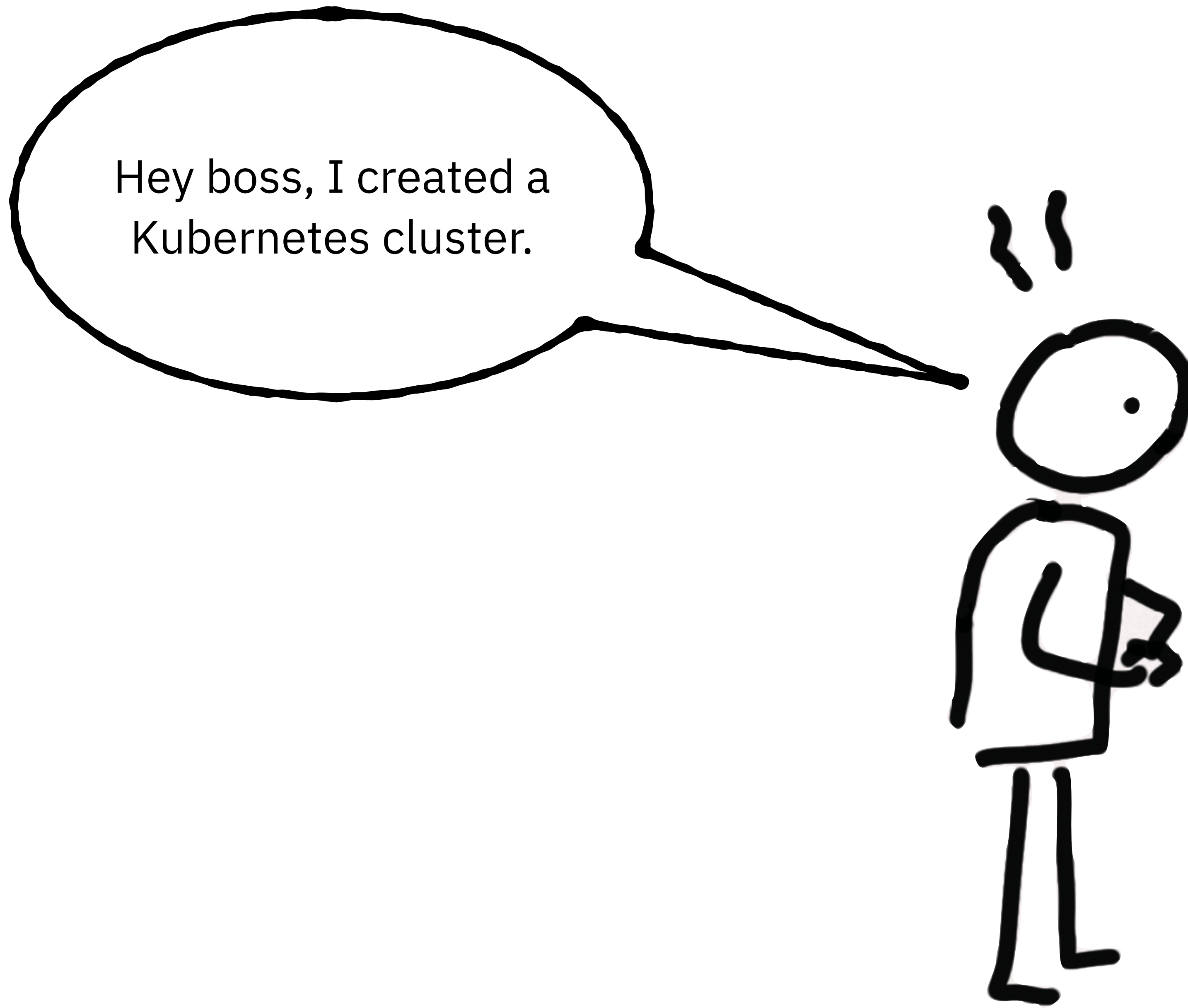
That doesn't mean the hardware  
is free.

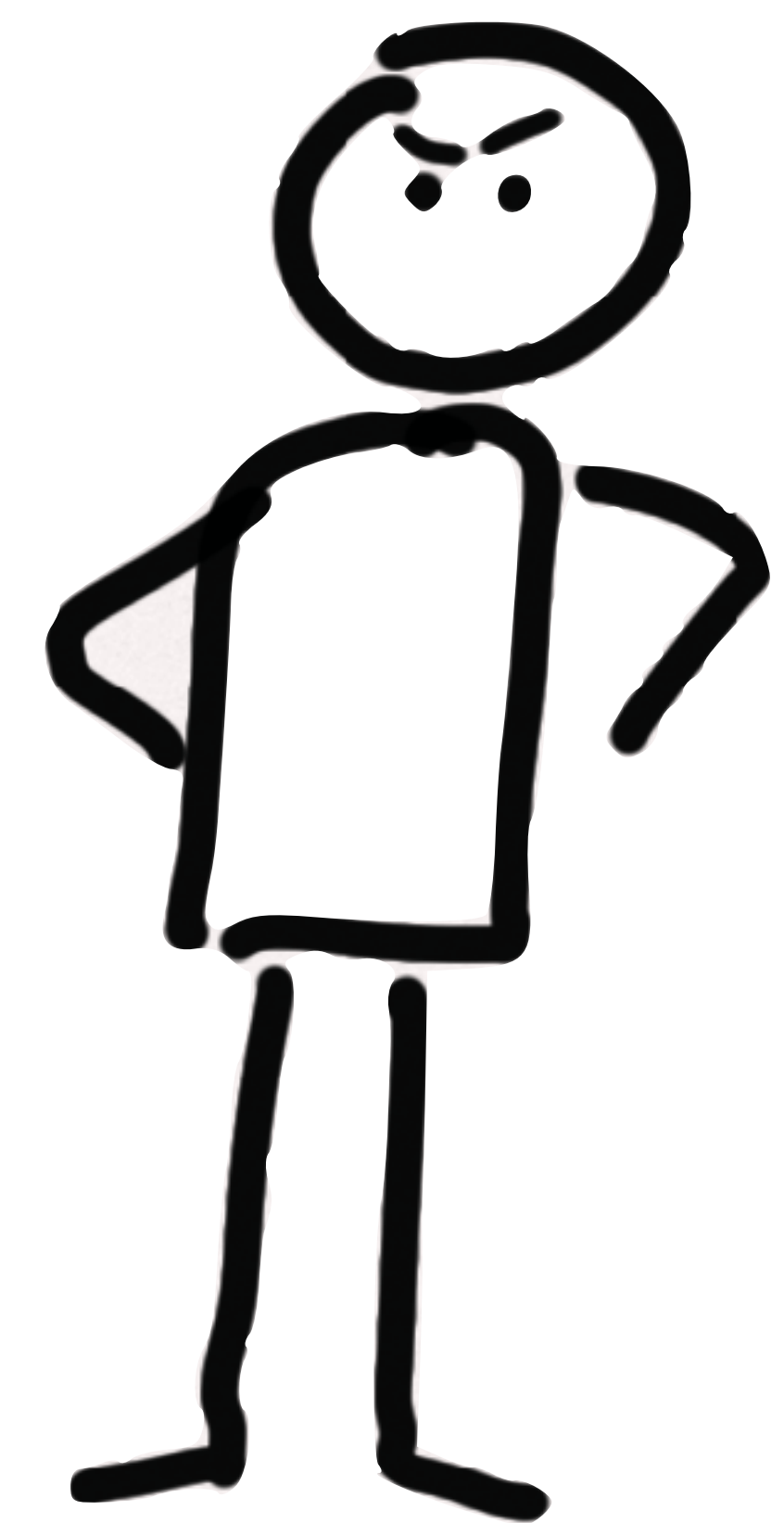
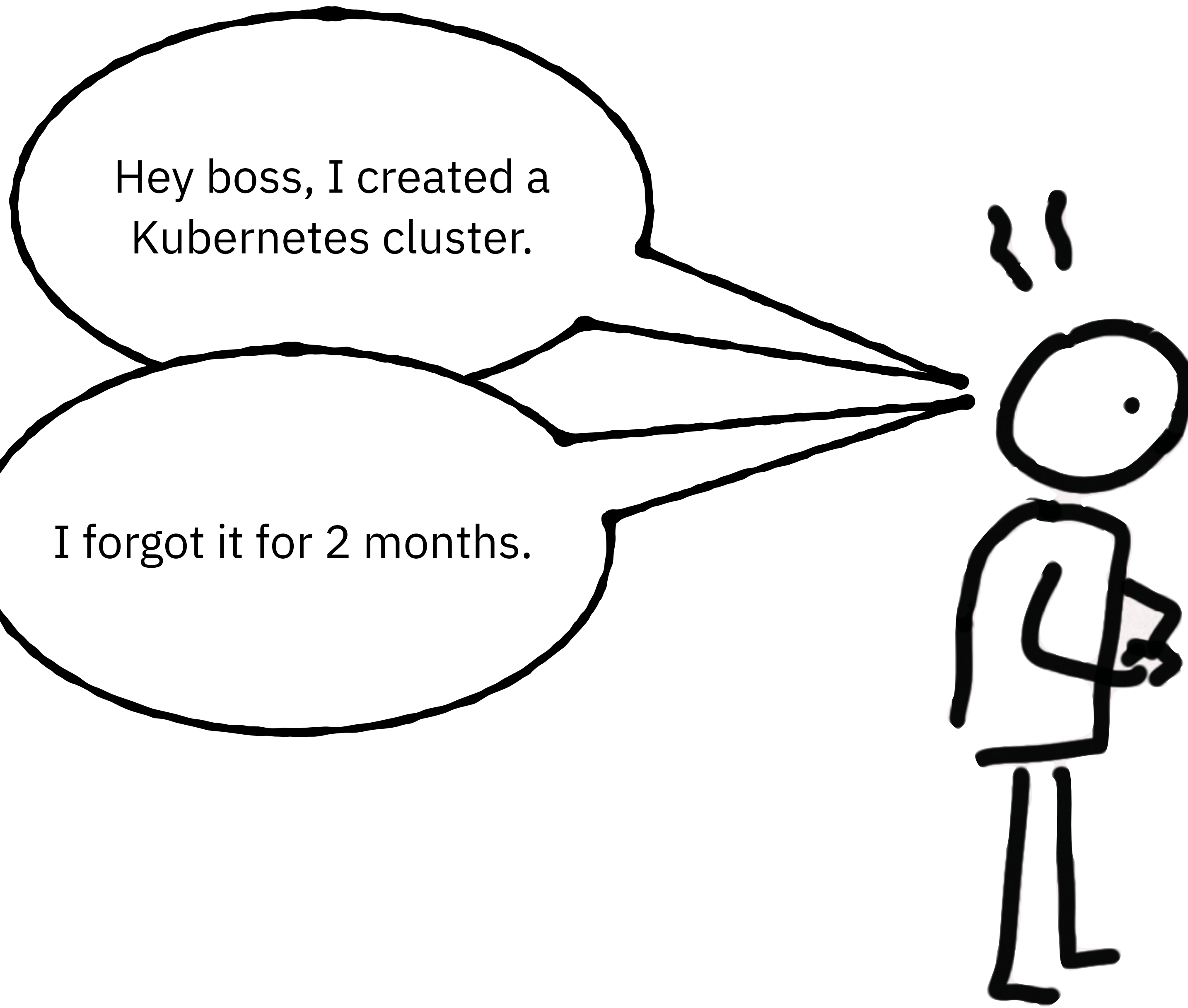
Or useful.

That doesn't mean the hardware  
is free.

Or useful.

Or shuts itself off.







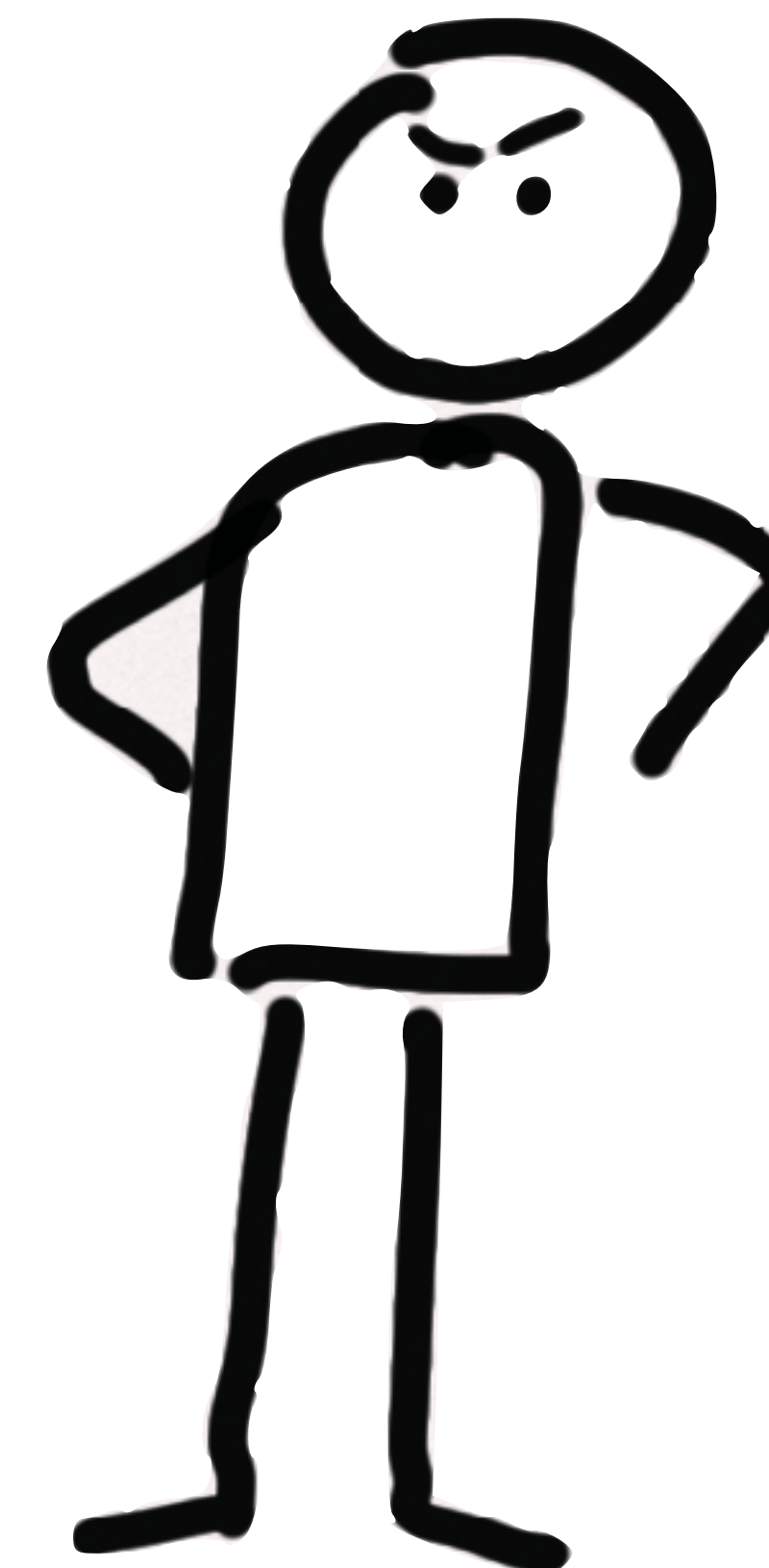
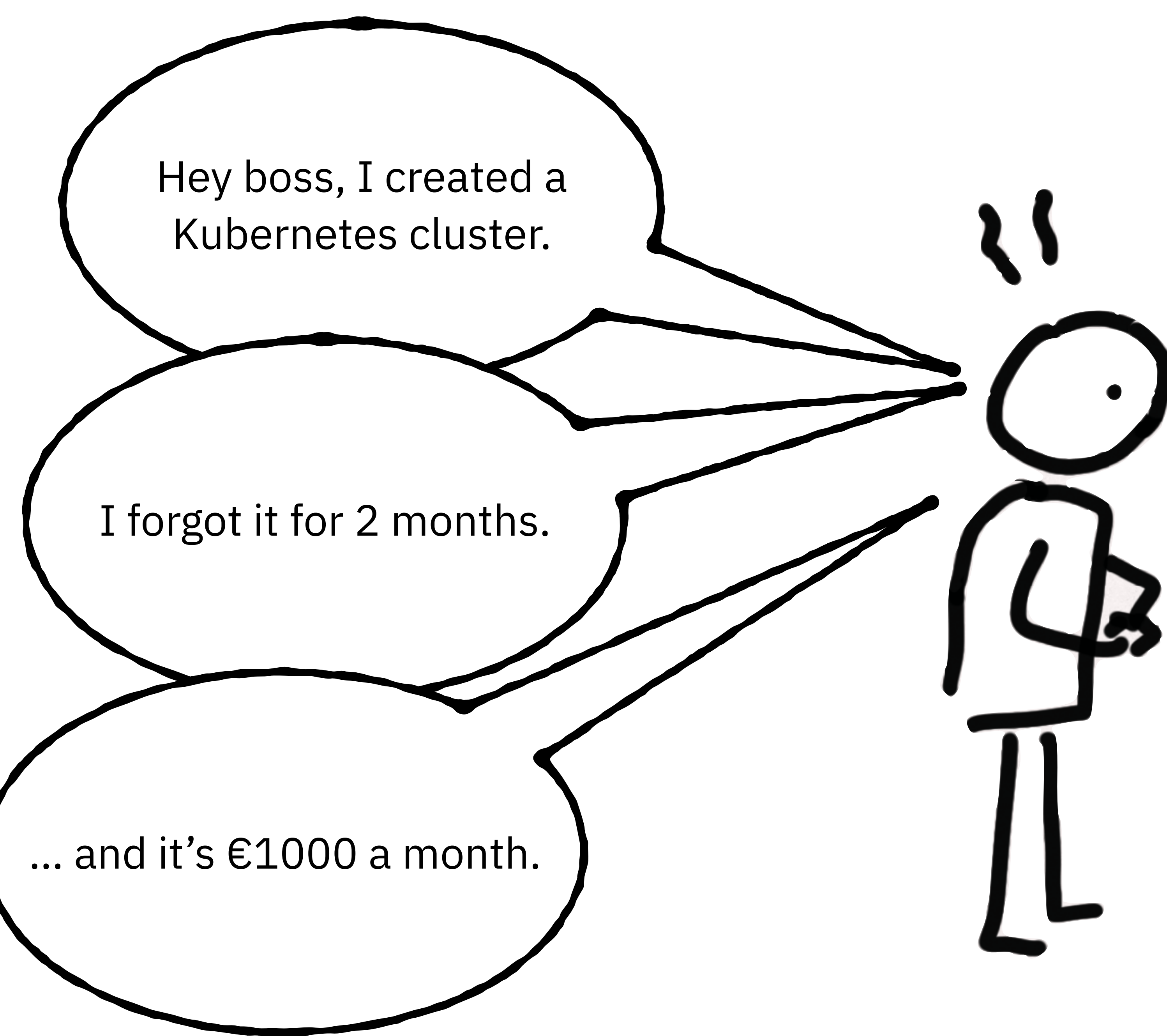






Photo by daveynin, flickr



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”





# finops

figuring out who in  
your company forgot  
to turn off their cloud

things that are  
different in the  
cloud

releasing





things that are  
different in the



# releasing





“we’re moving too slowly.



“we’re moving too slowly.

we should modernise our COBOL  
application into microservices.

“we’re moving too slowly.

we should modernise our COBOL  
application into microservices.

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

# modularity

microservices are not the **goal**

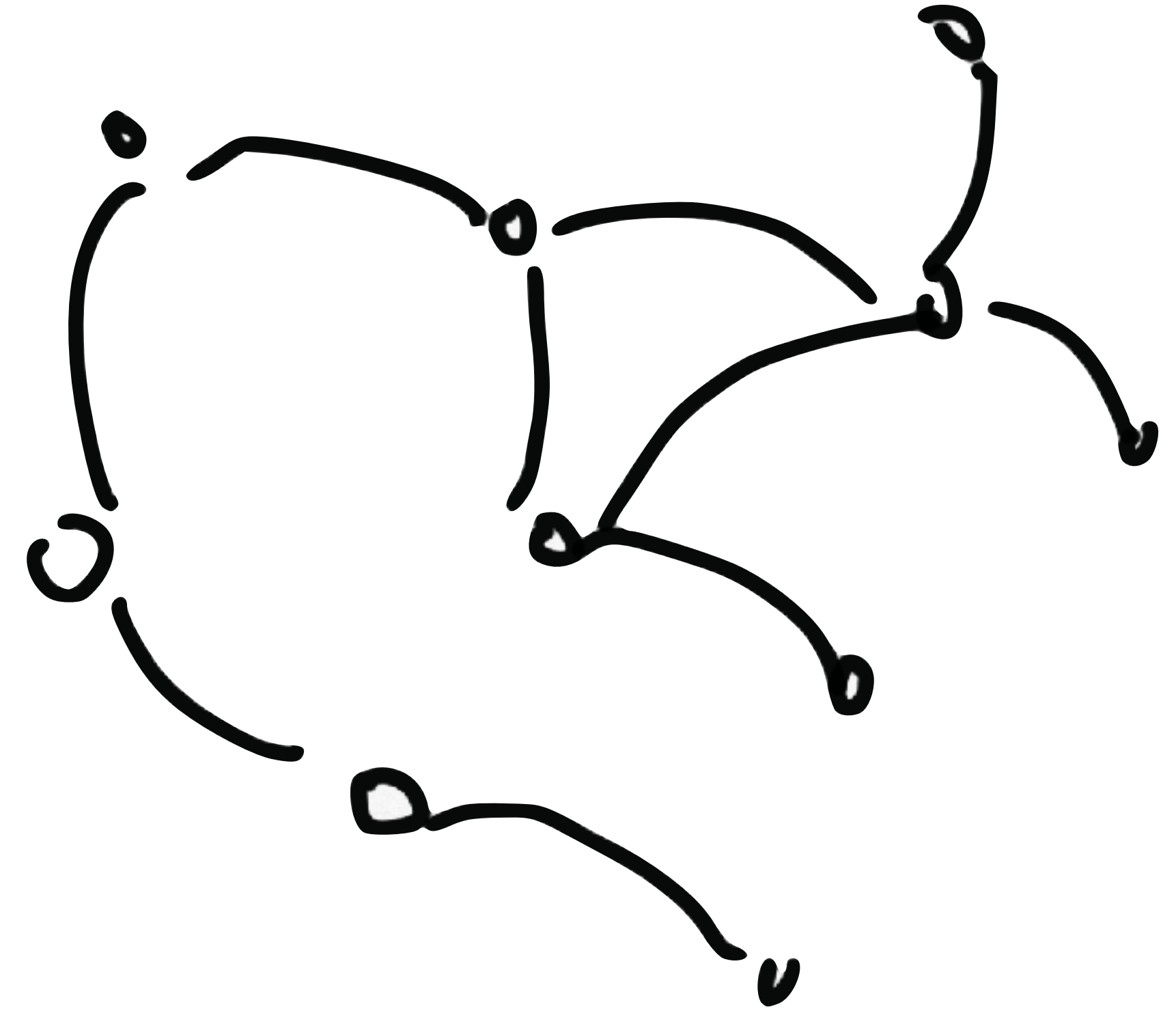
microservices are not the **goal**  
they are the **means**



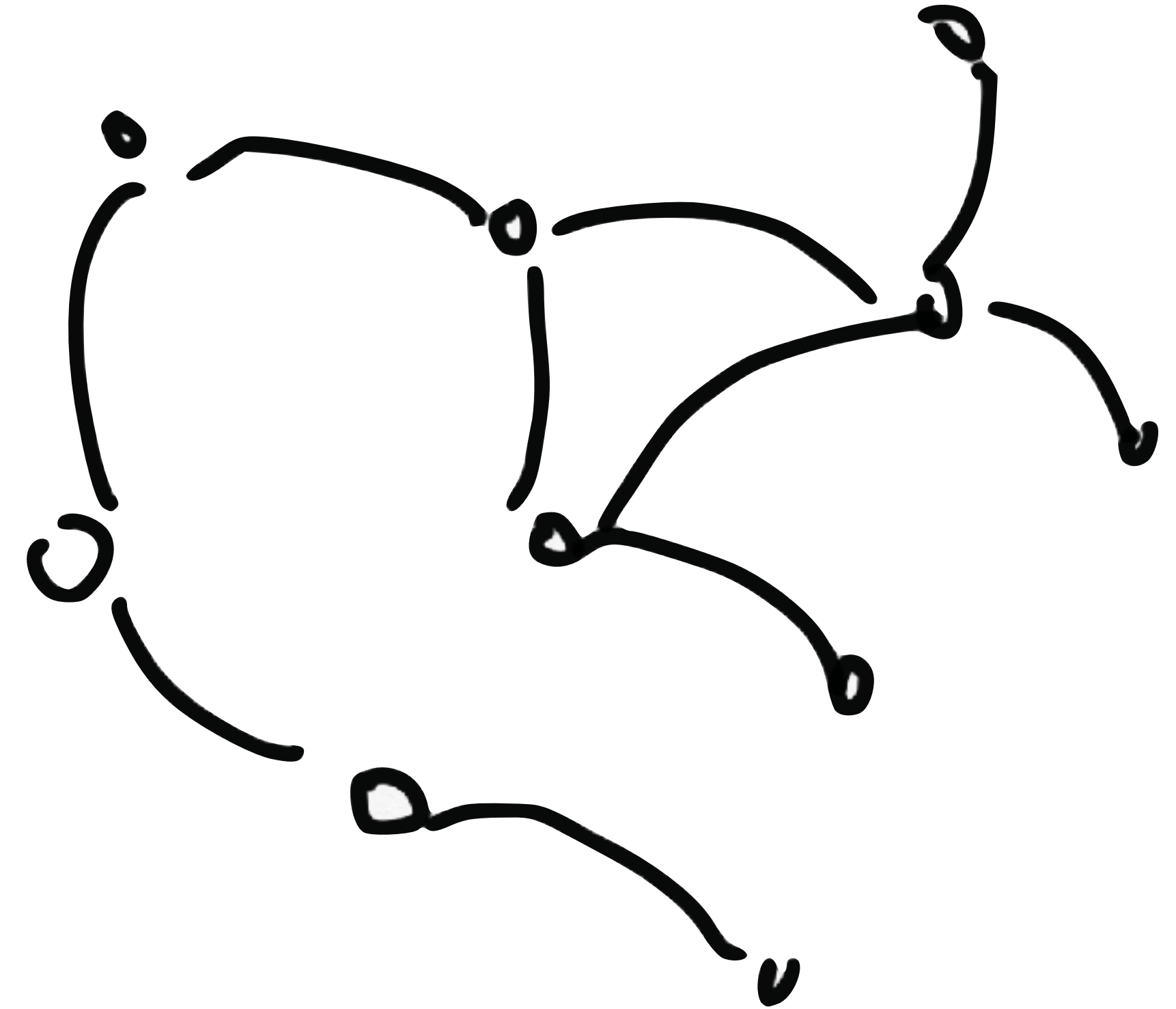
wishful mimicry

cloud != microservices

“every time we touch one  
microservice, all the others break.”



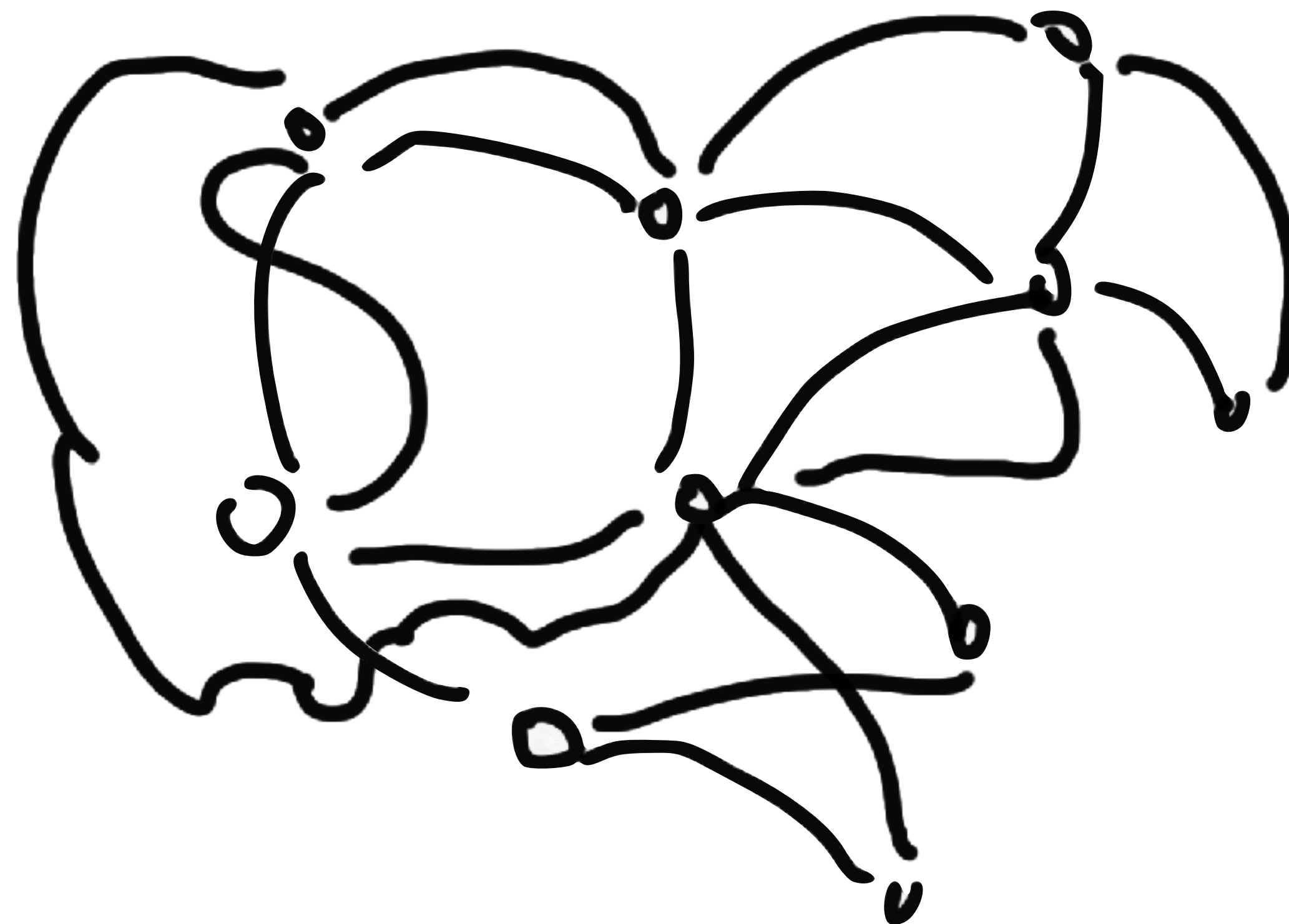
# distributed monolith



# distributed monolith

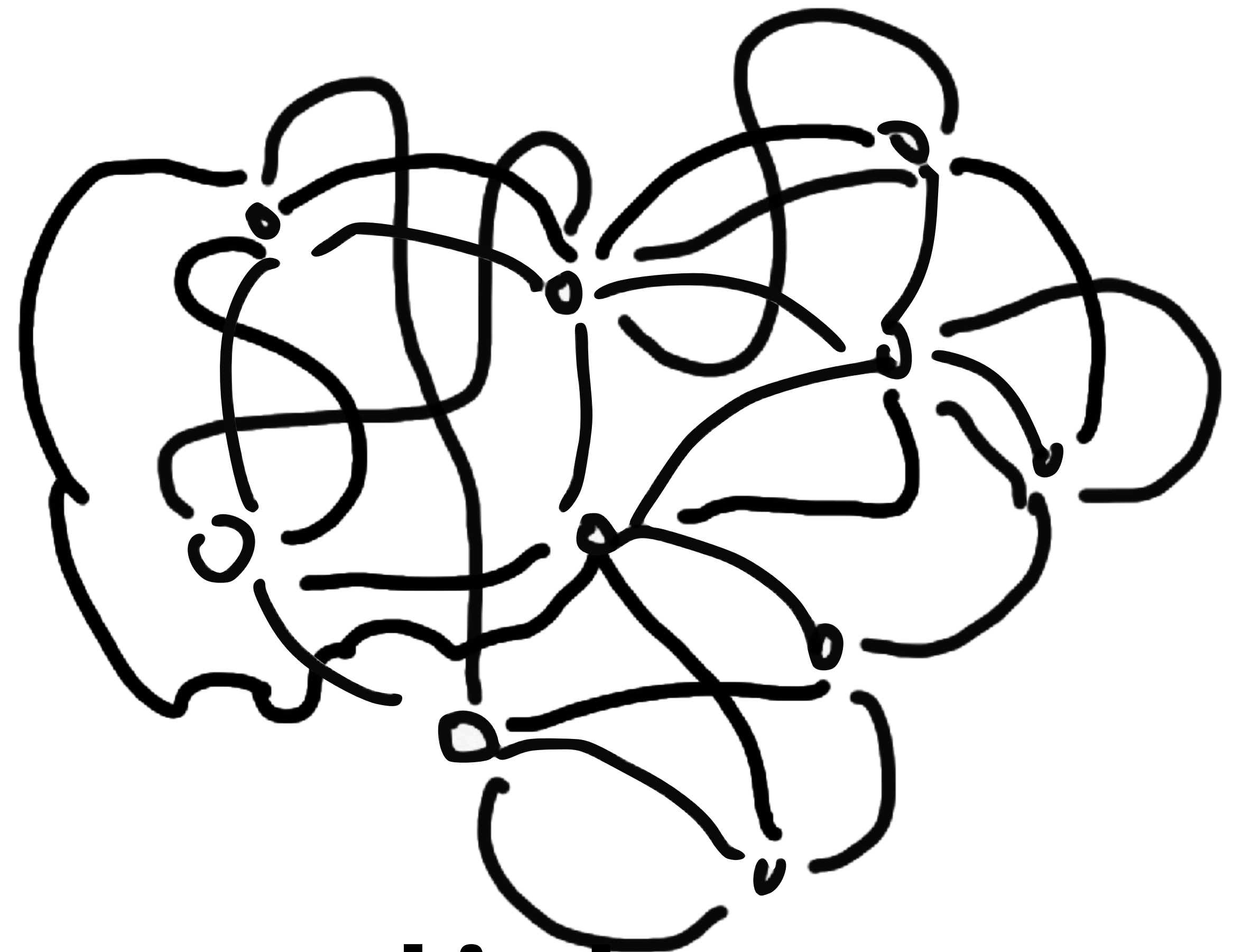
but without compile-time checking  
... or guaranteed function execution





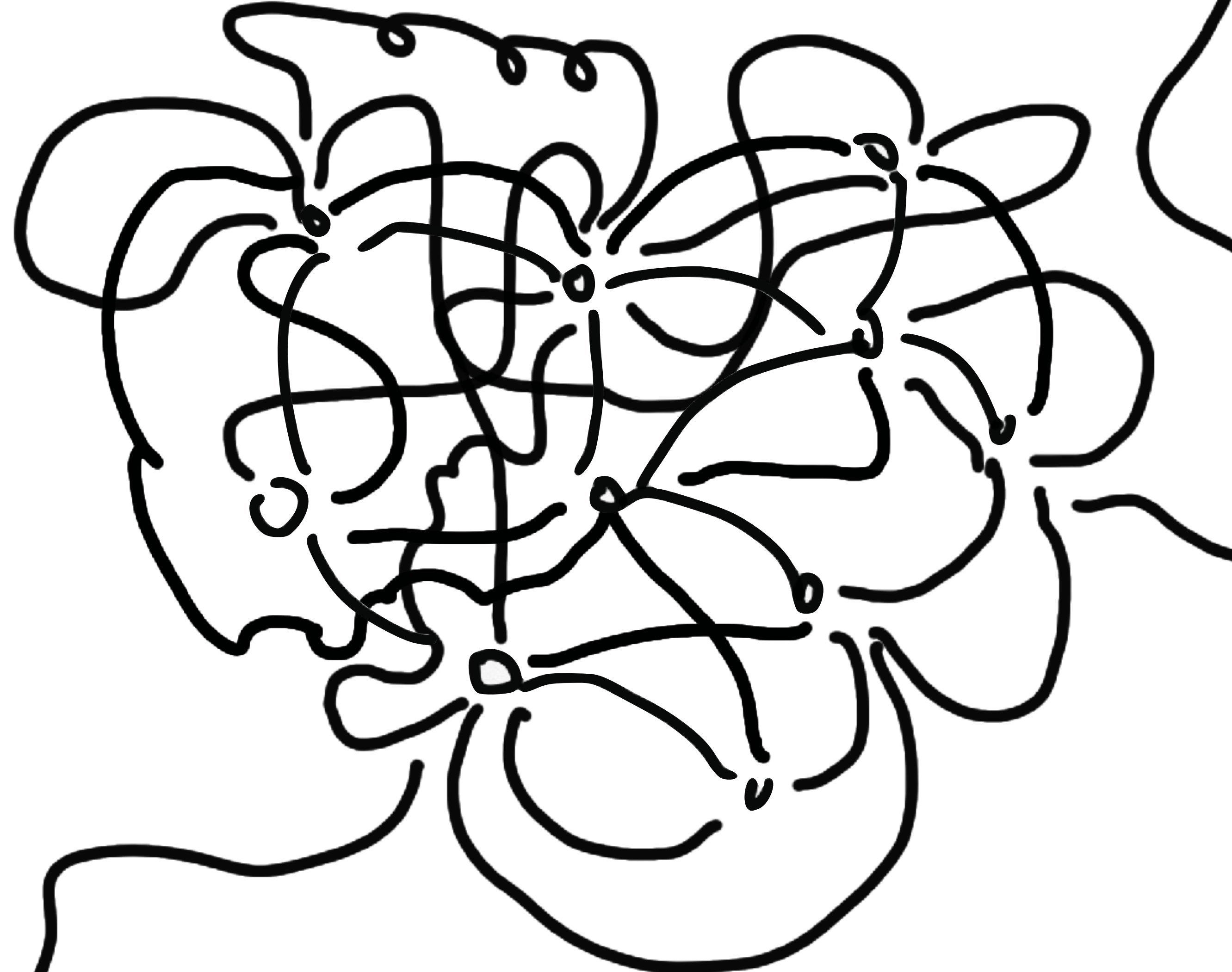
# distributed monolith

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



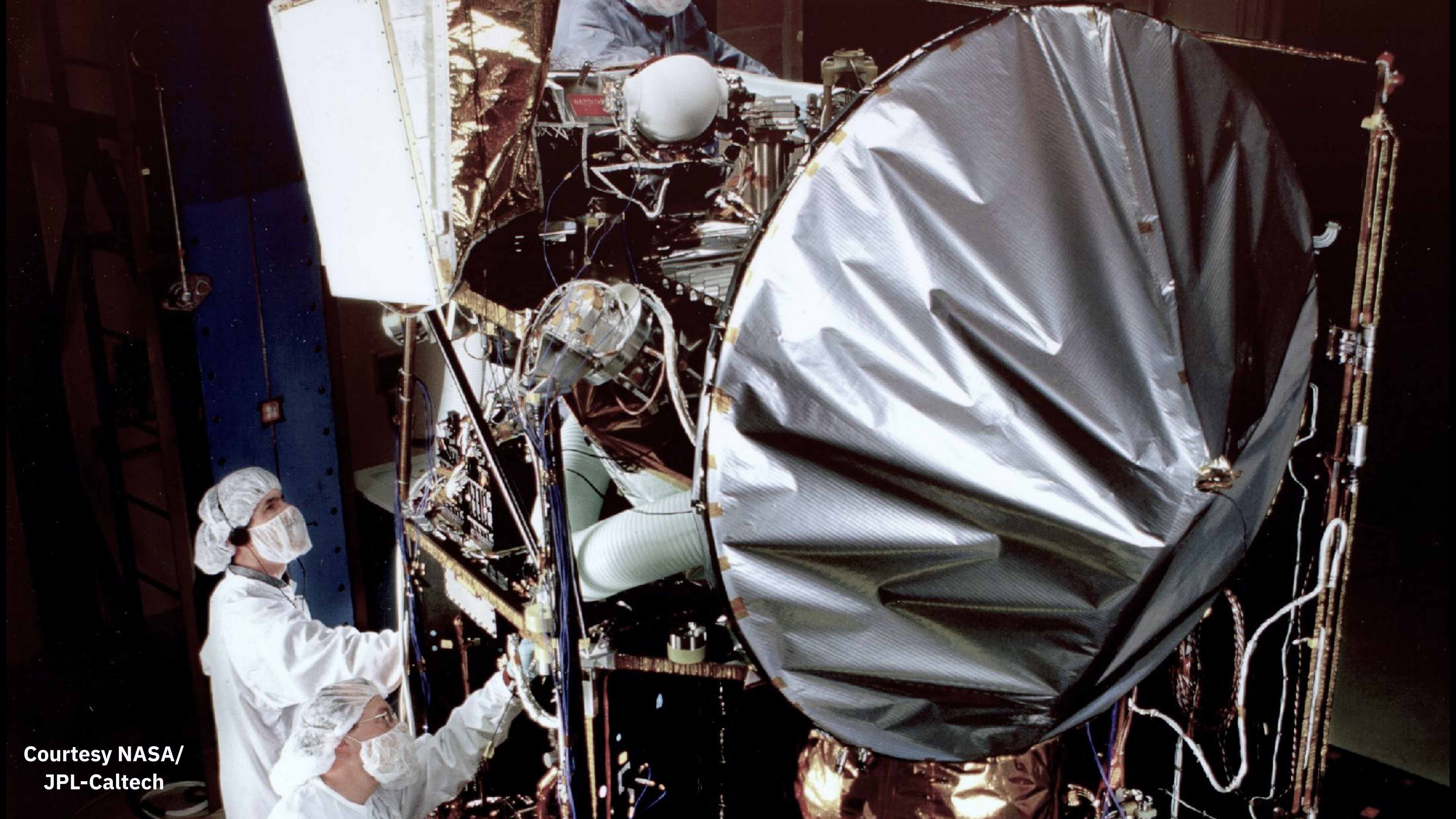
# distributed monolith

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



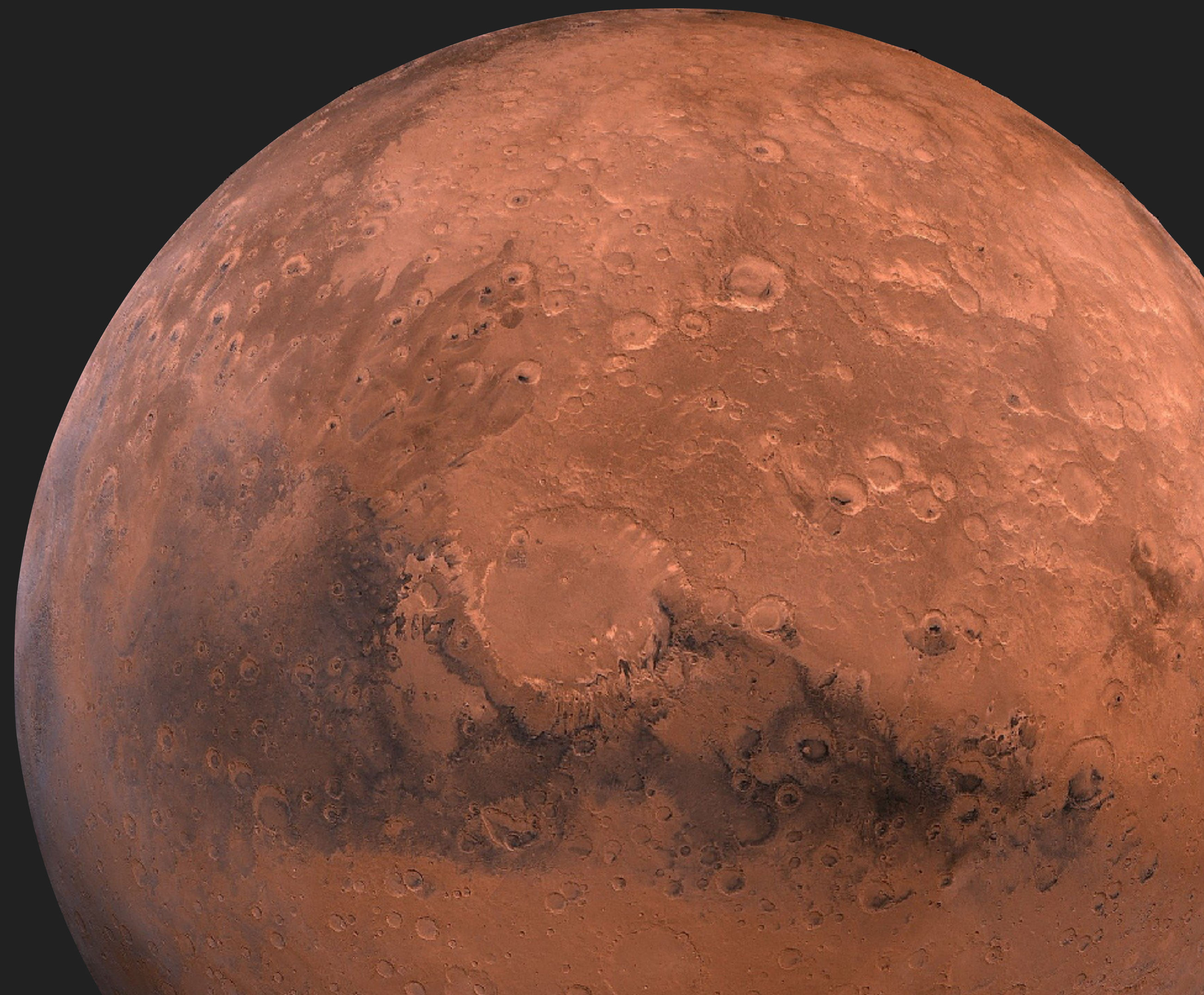
distributed monolith  
but without compile-time checking  
... or guaranteed function execution



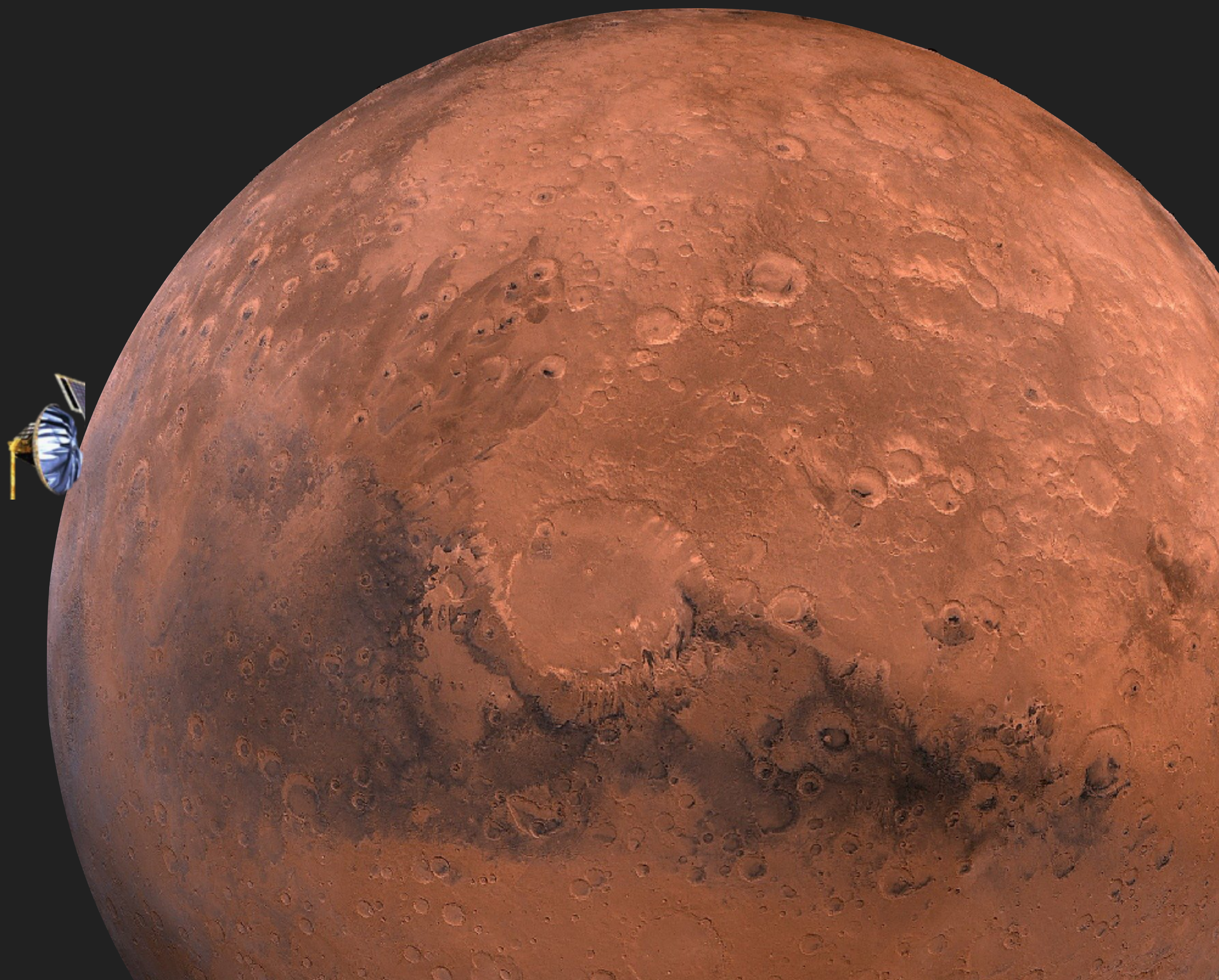


Courtesy NASA/  
JPL-Caltech

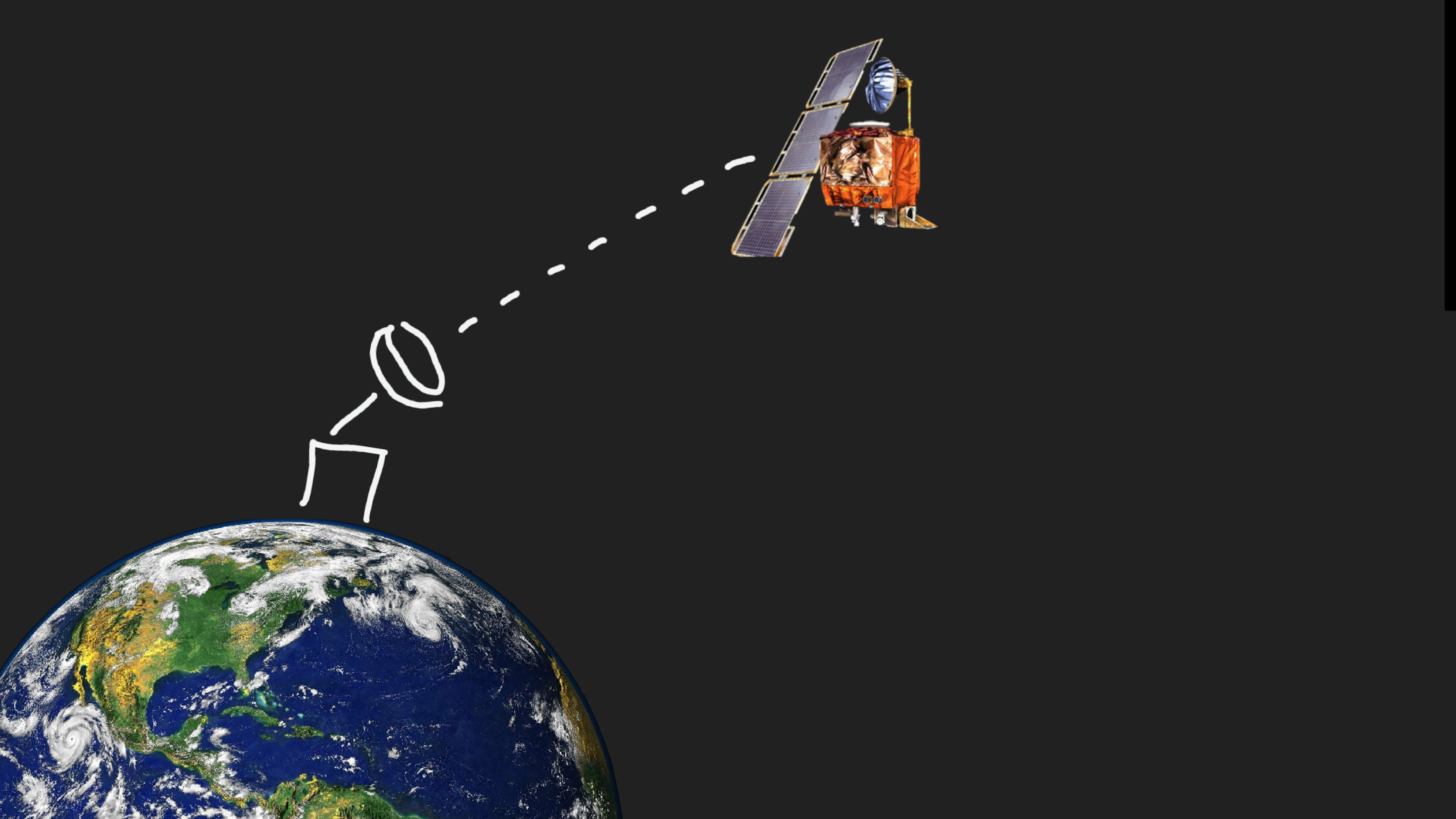




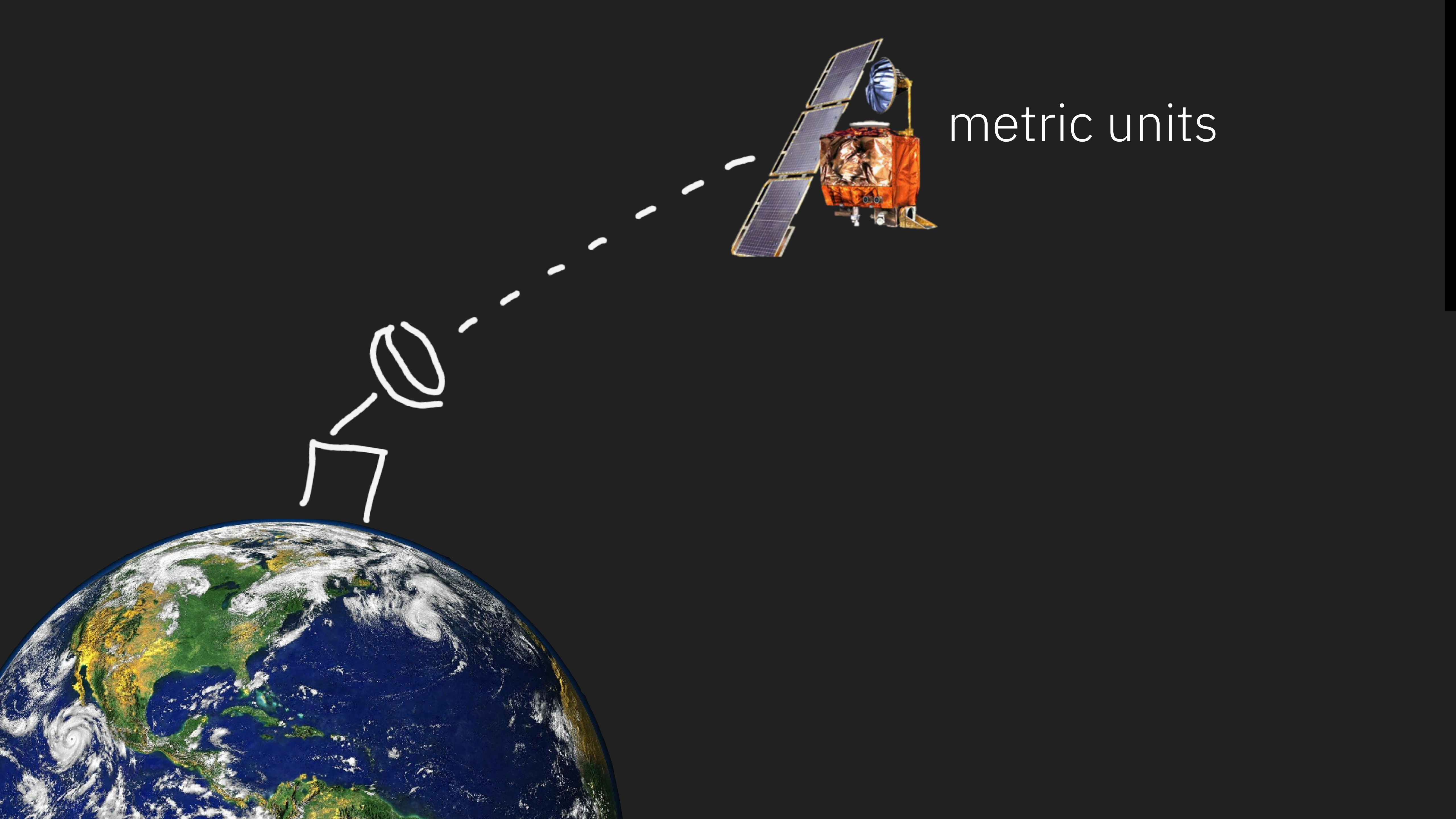






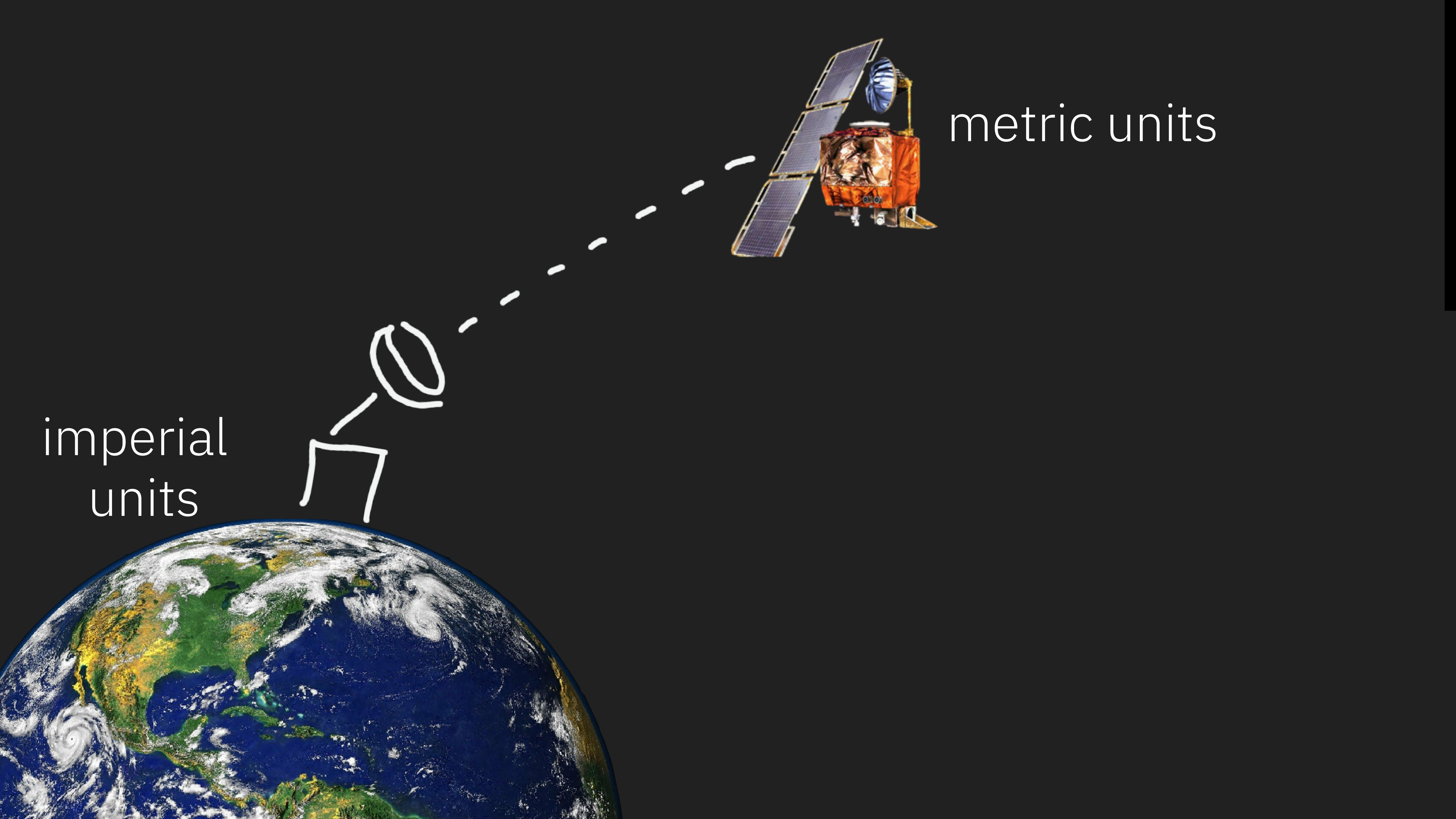






metric units





imperial  
units

metric units





metric units

imperial  
units

distributing  
did not help

distributed  $\neq$  decoupled



# reasons **not** to do microservices

small team

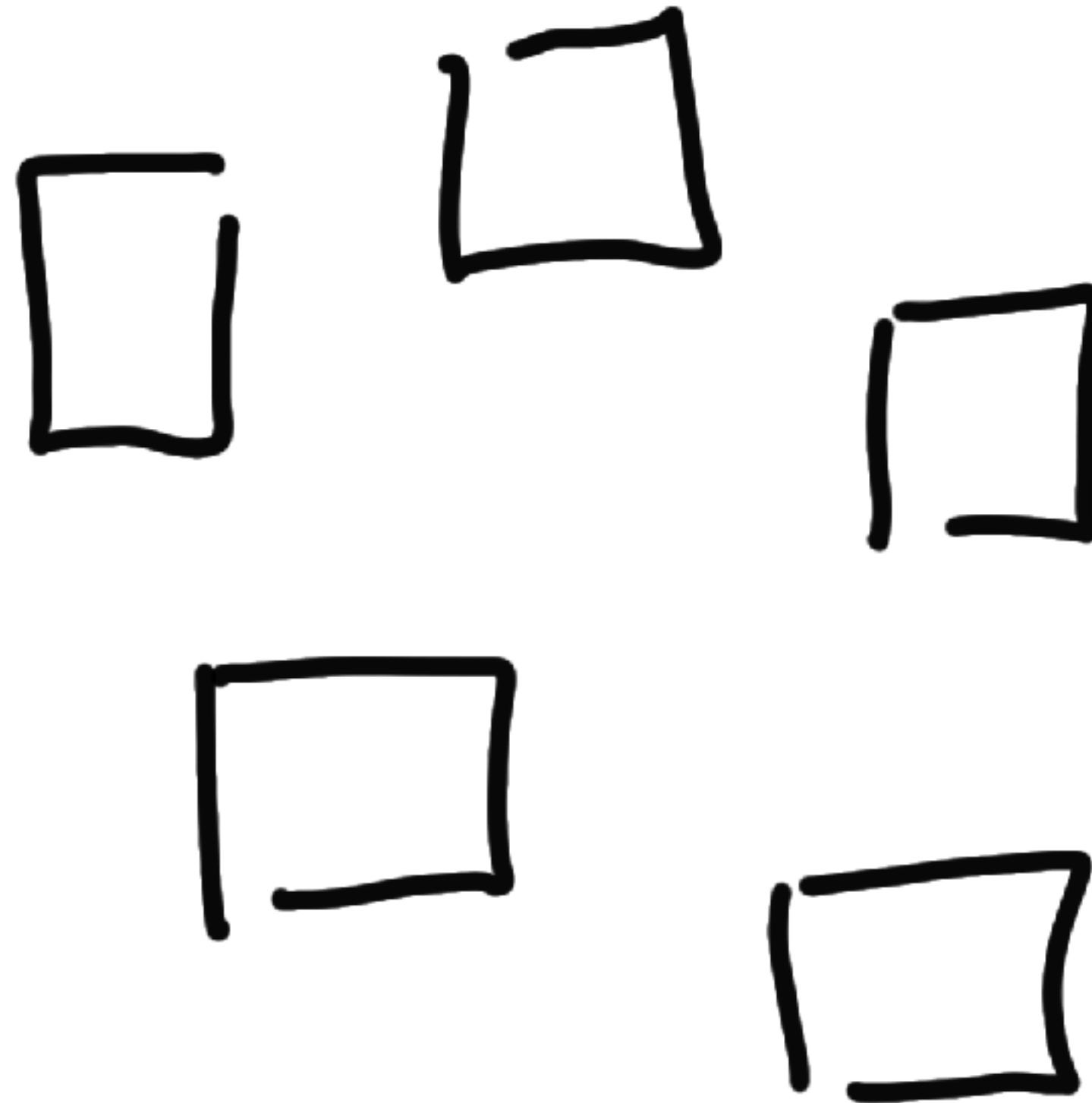
not planning to release independently

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

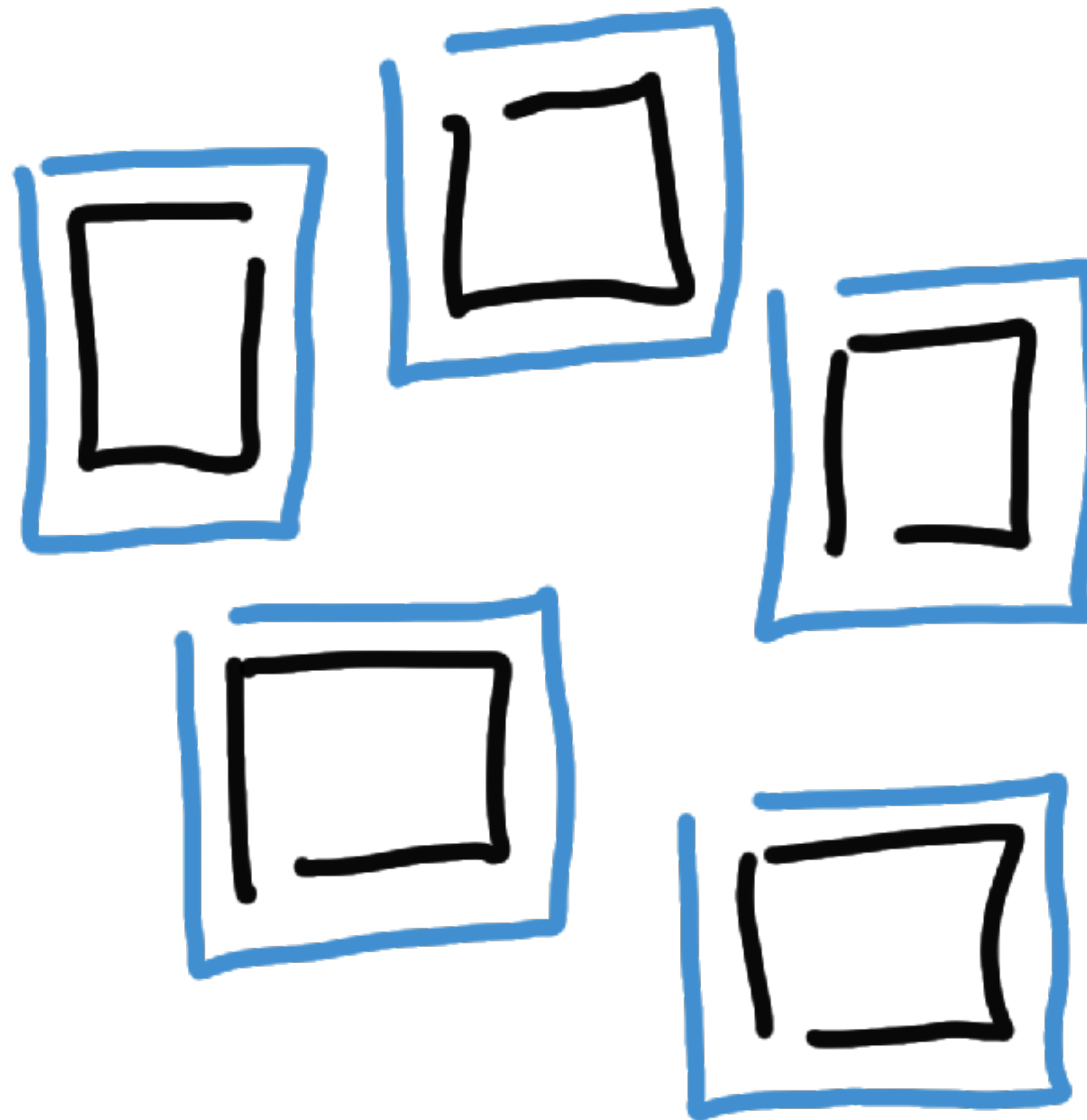
domain model doesn't split nicely

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

~ **Microservice**

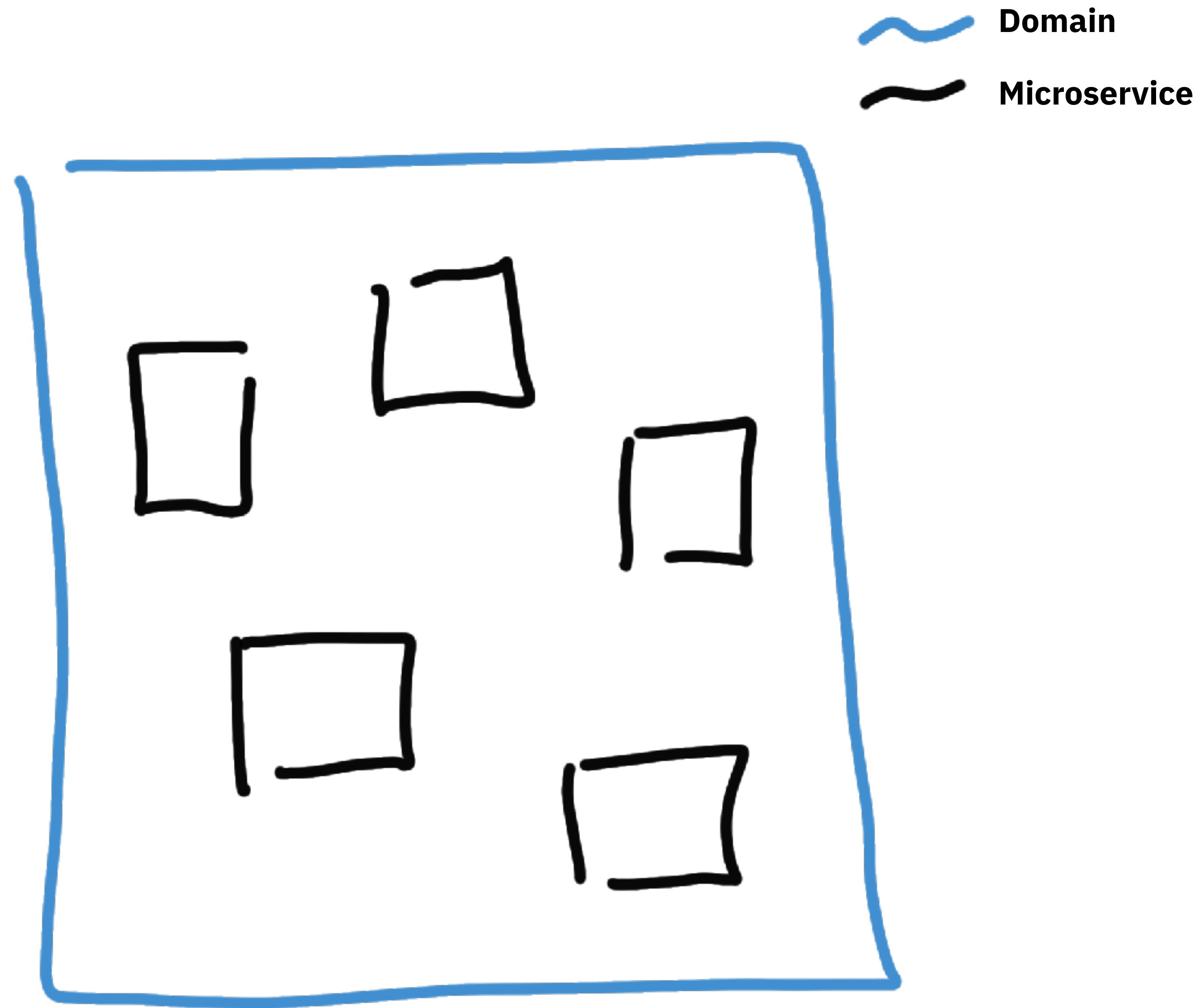


Domain  
Microservice





(this is bad)



If you're going to do microservices,  
you need to get good at automation.

**And testing.**







the test  
pyramid





the test  
pyramid





the test  
pyramid





the test  
pyramid





# the test pyramid

(you can TDD  
at every level!)





# How to test a fire alarm?





how **not** to test a fire alarm





how **not** to test a fire alarm







unit testing a fire  
alarm





uh ... is that enough?





contract testing a  
fire alarm





contract testing a  
fire alarm



# to the **code!**

Public  
class  
private  
private  
this  
author  
...



microservices **need**  
consumer-driven contract tests





**thank you**

@holly\_cummins

**get the code:**

[ibm.biz/holly-jfokus](https://ibm.biz/holly-jfokus)