

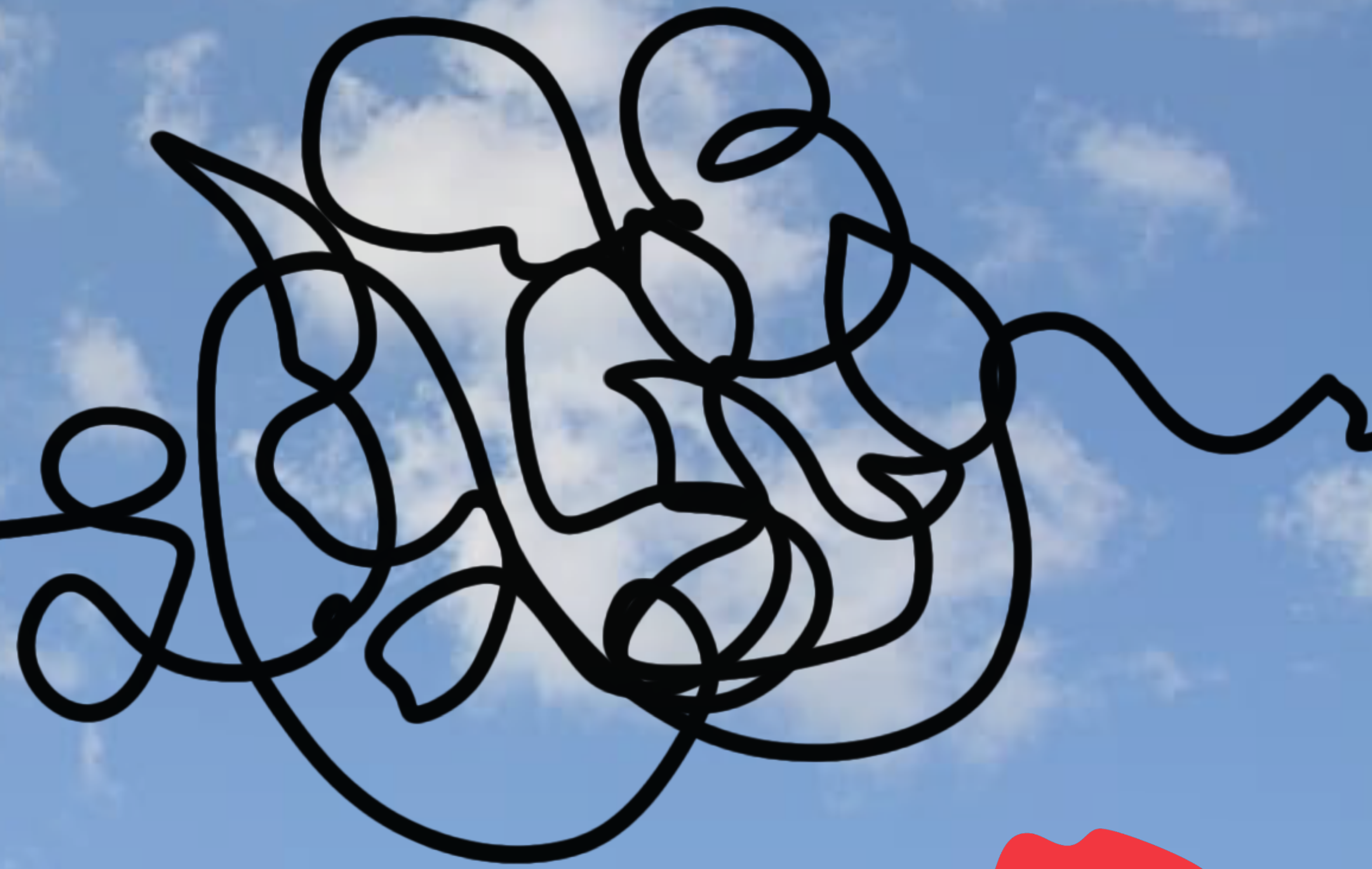
goto;

GOTO Copenhagen 2022

#GOTOCph

cloud chaos and microservices mayhem

Holly Cummins
Red Hat
@holly_cummins







**things you
need to do
well in 2022:**

handwashing

**things you
need to do
well in 2022:**

handwashing
apps

**things you
need to do
well in 2022:**

**things you
need to do
well in 2022:**

handwashing
apps
ops

**things you
need to do
well in 2022:**

handwashing

apps

ops

devops

**things you
need to do
well in 2022:**

handwashing

apps

ops

devops

devsecops

**things you
need to do
well in 2022:**

handwashing

apps

ops

devops

devsecops

finops





a few
things that are
different in the





technology changes fast (duh)



technology changes fast (duh)


... but it's taking us a while to catch up to cloud



technology changes fast (duh)

... but it's taking us a while to catch up to cloud


things that are
different in the



tracing



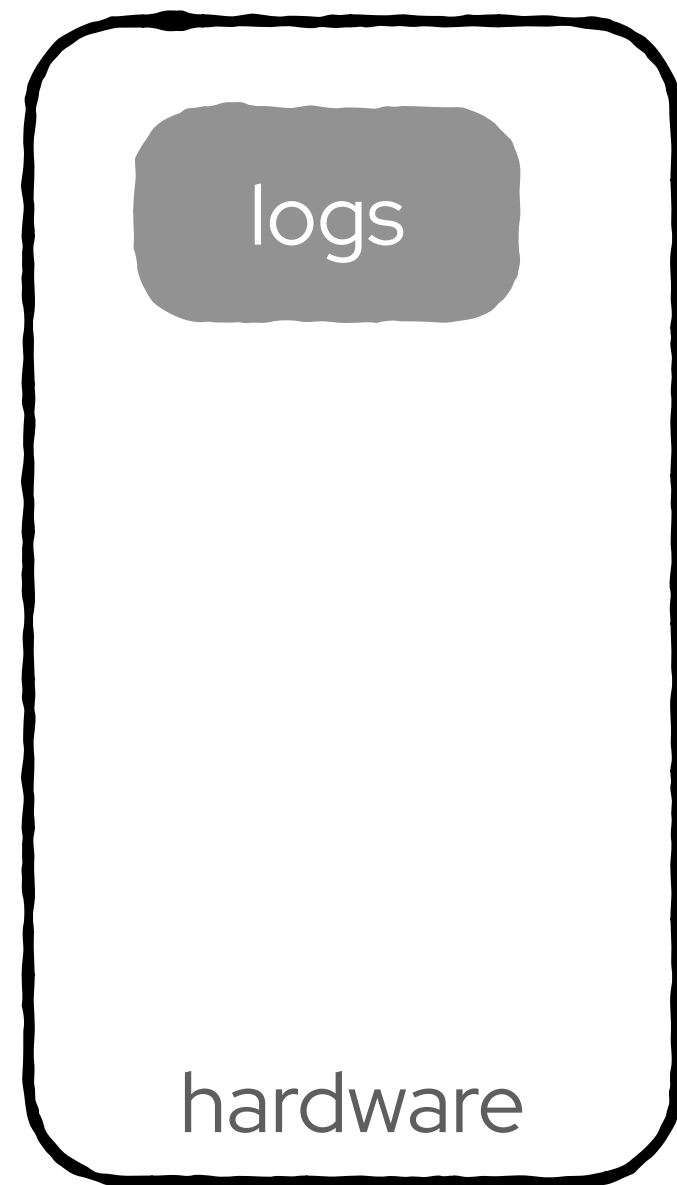
things that are
different in the



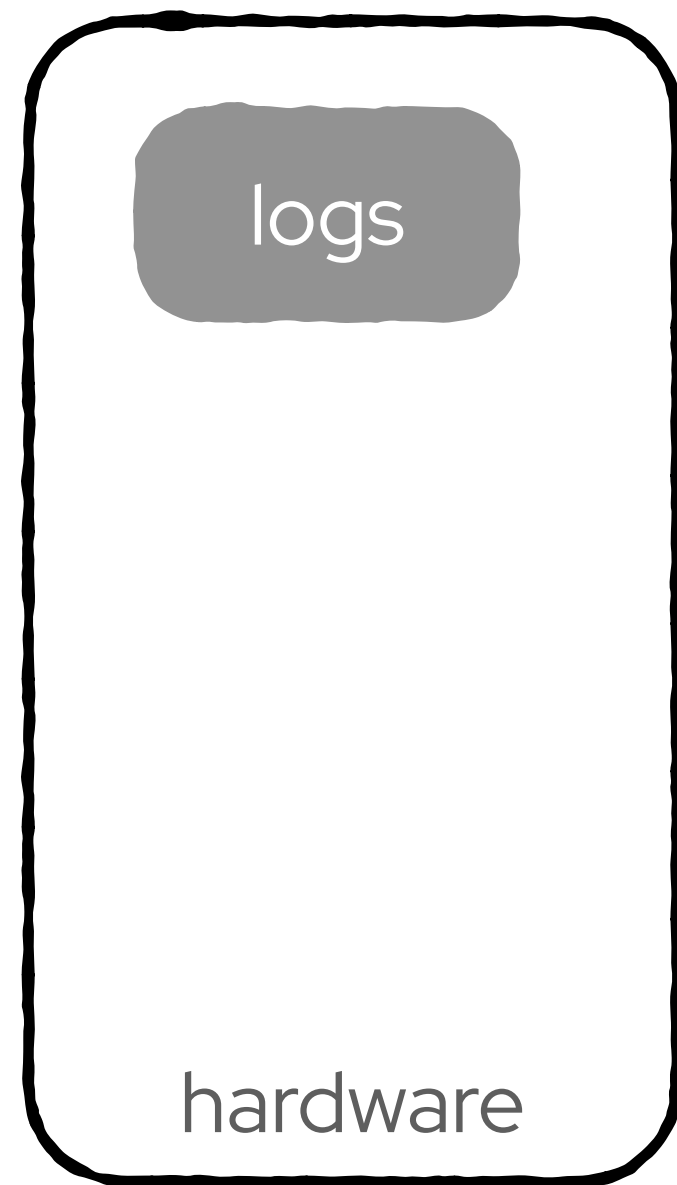
observability



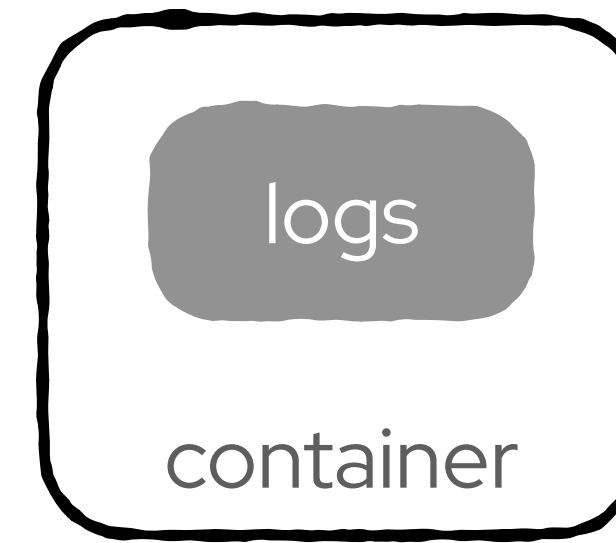
old way



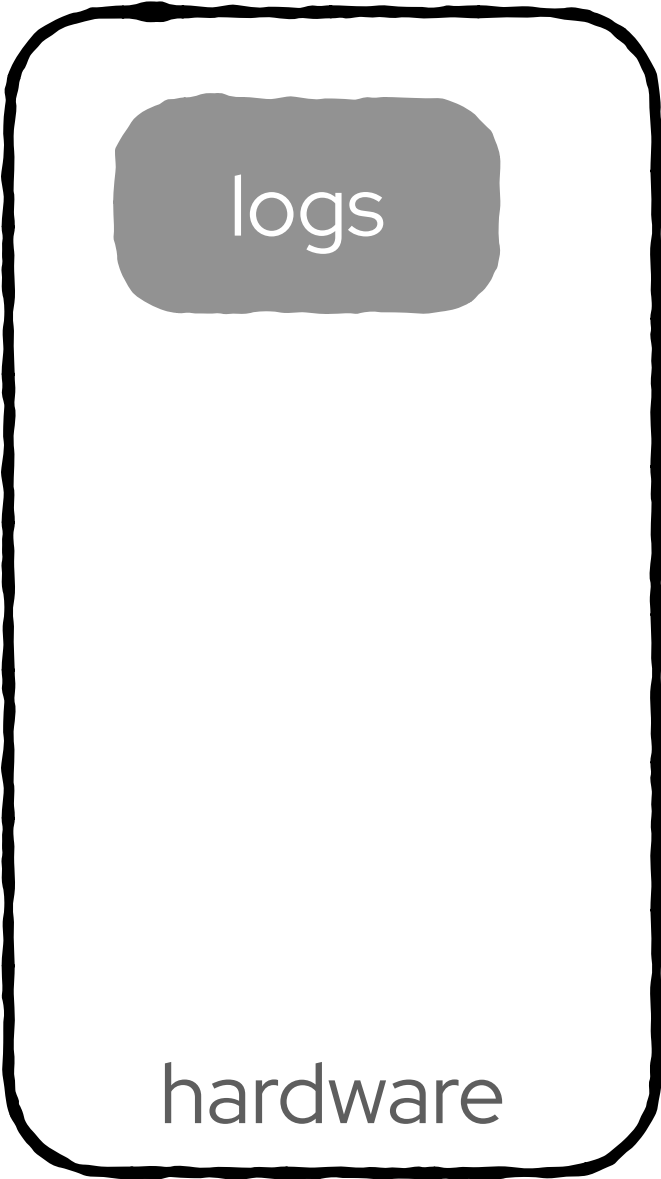
old way



cloud way



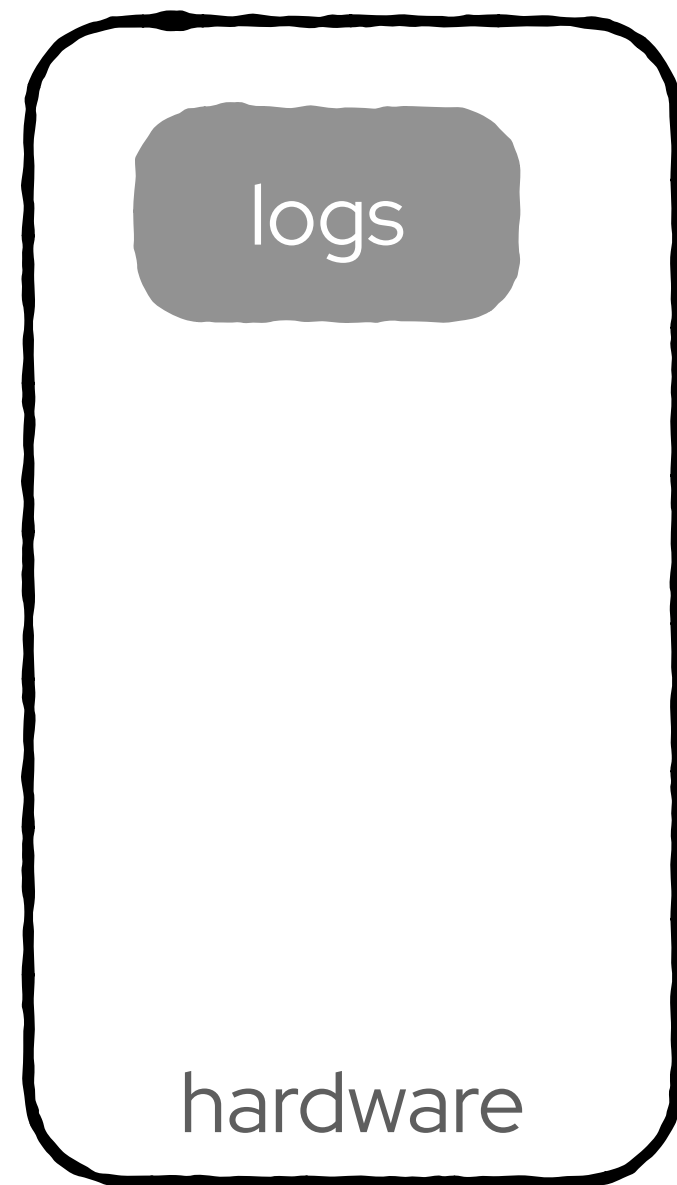
old way



cloud way



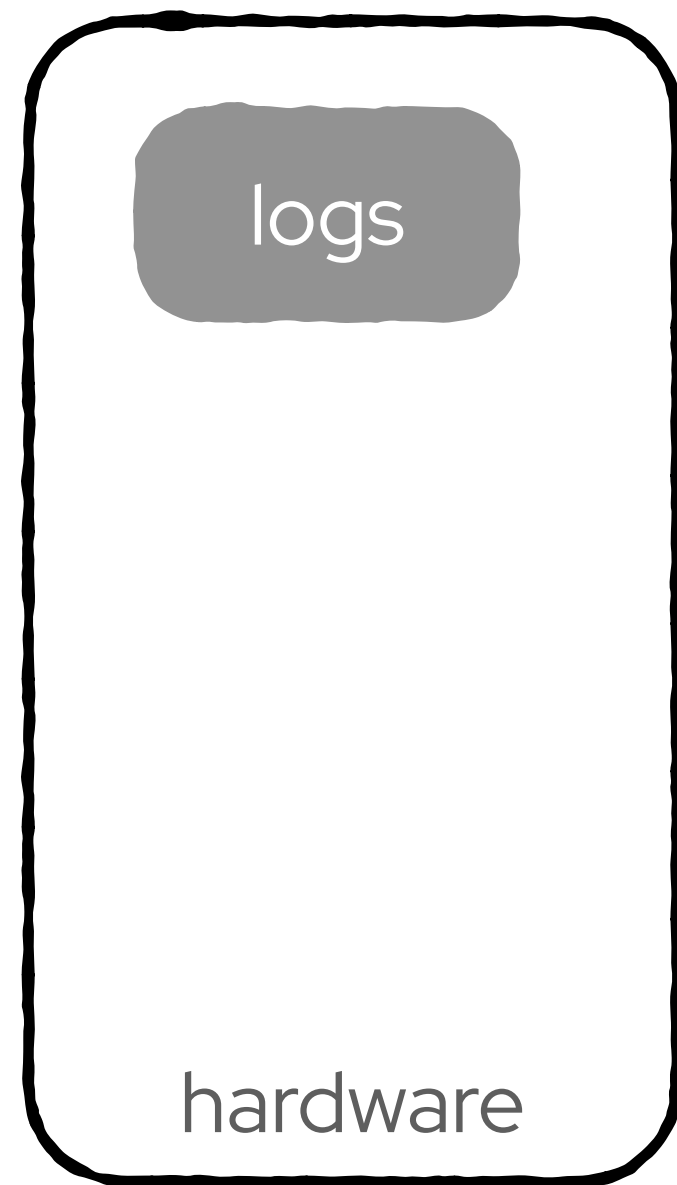
old way



cloud way

oops

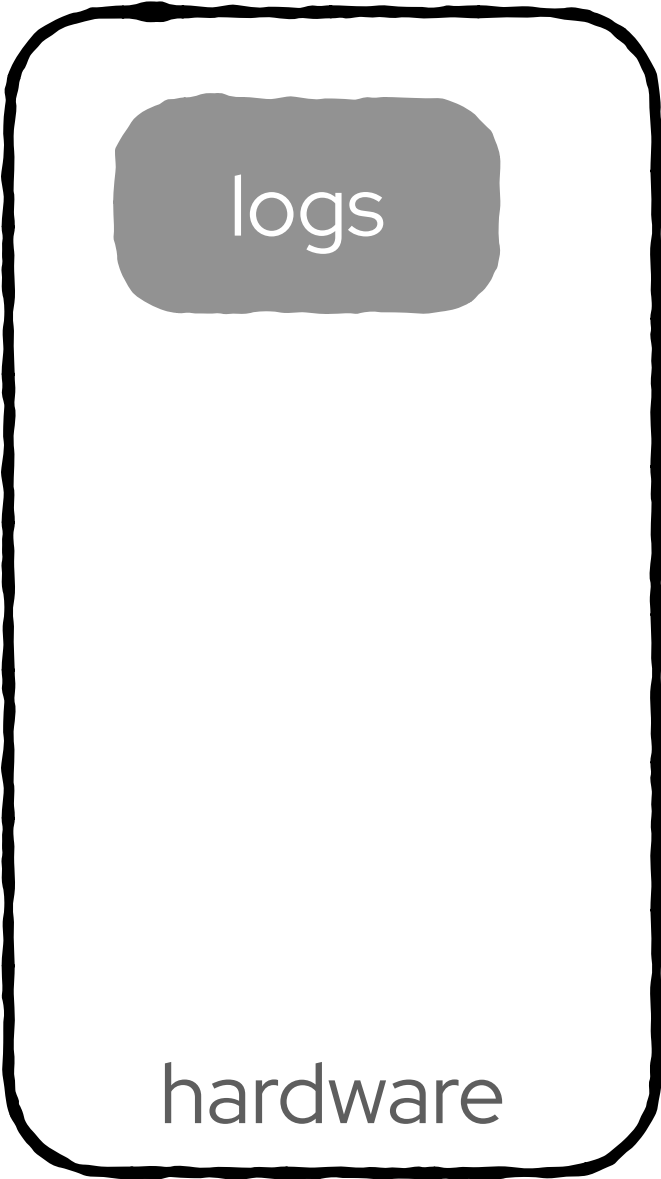
old way



cloud way



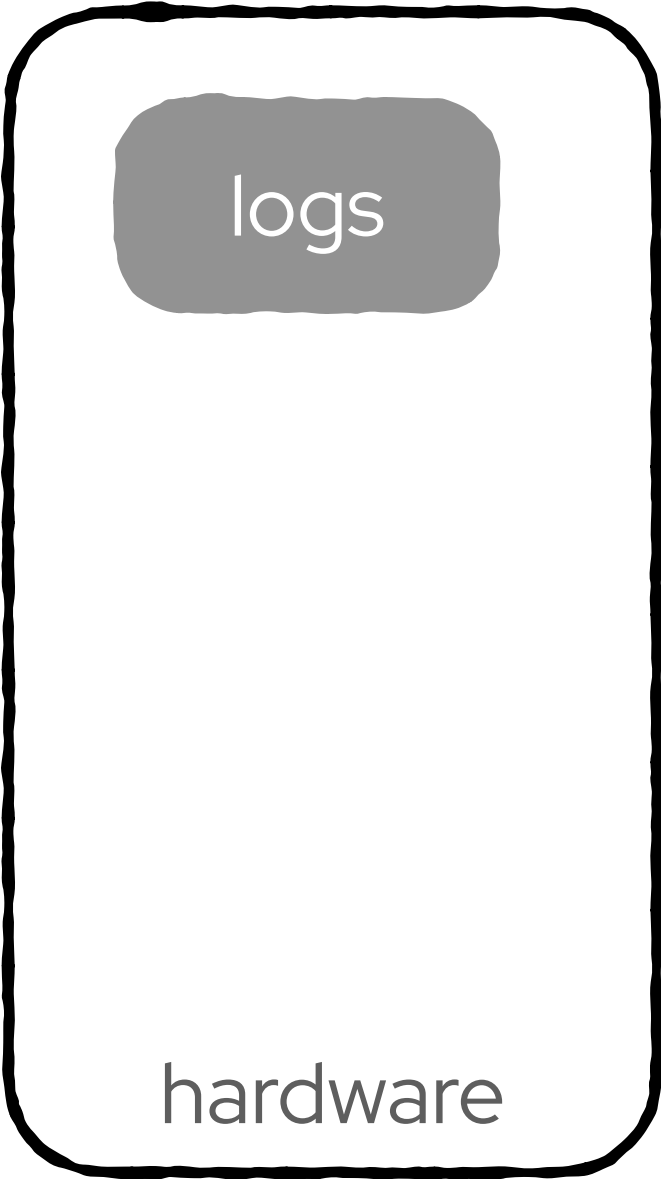
old way



cloud way



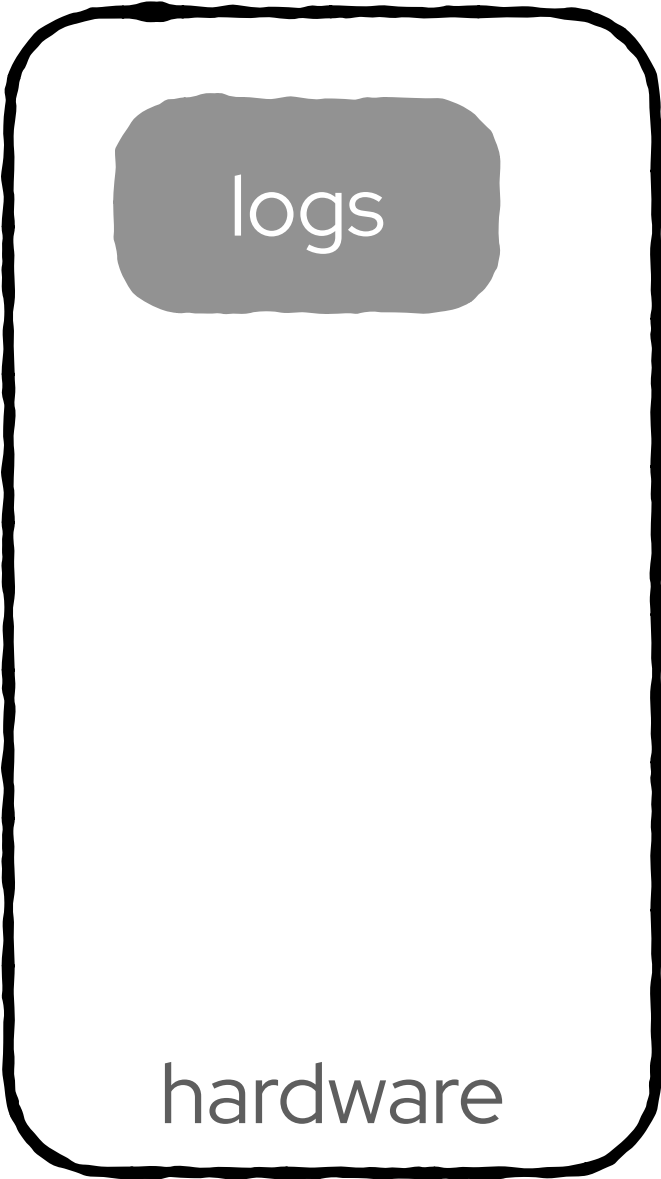
old way



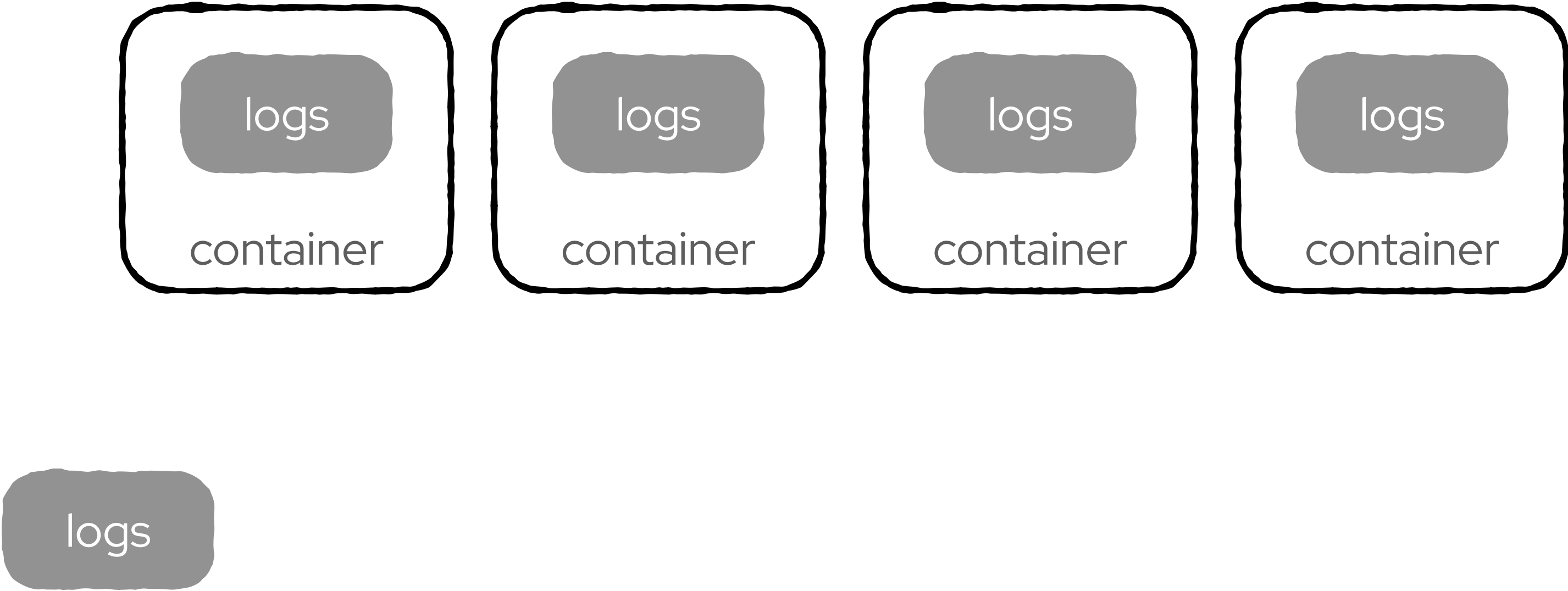
cloud way



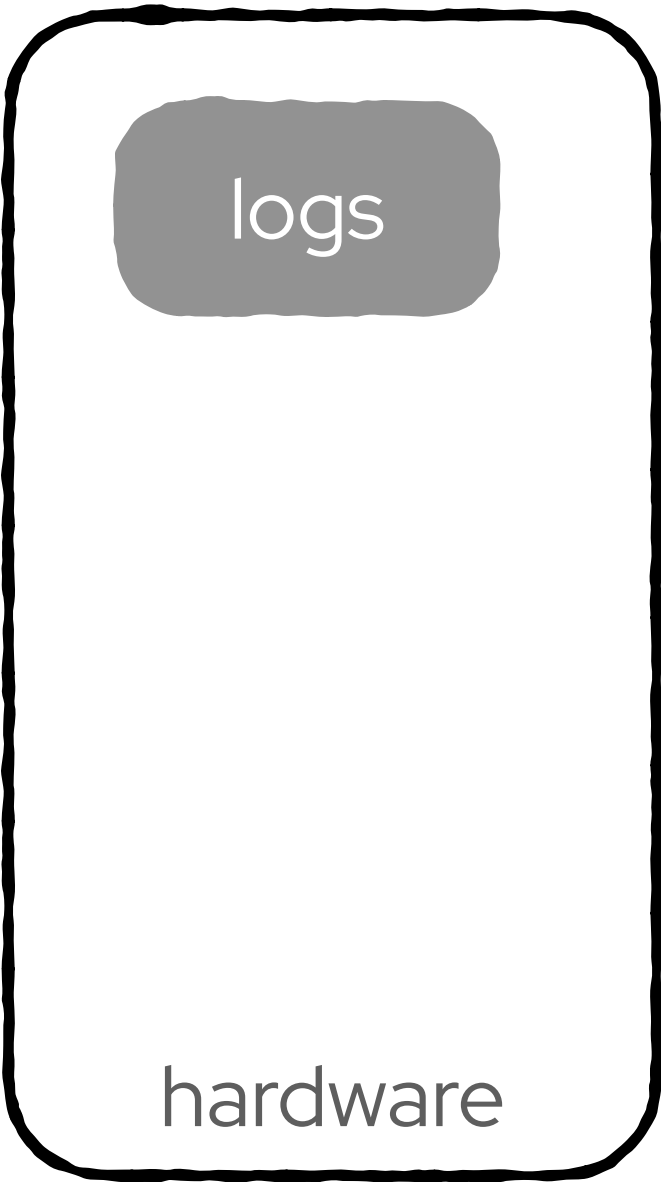
old way



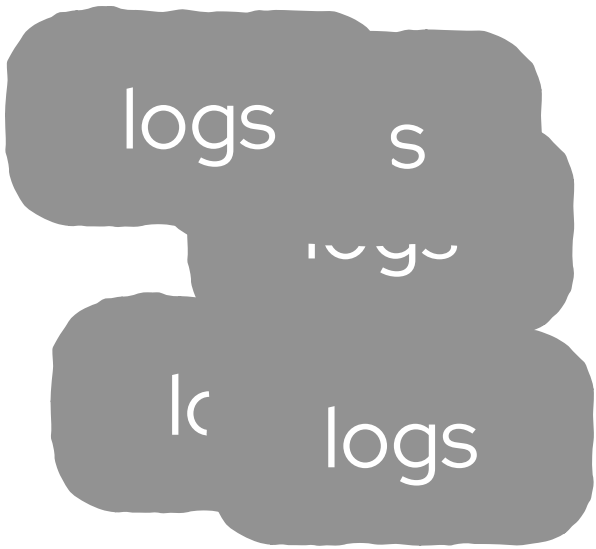
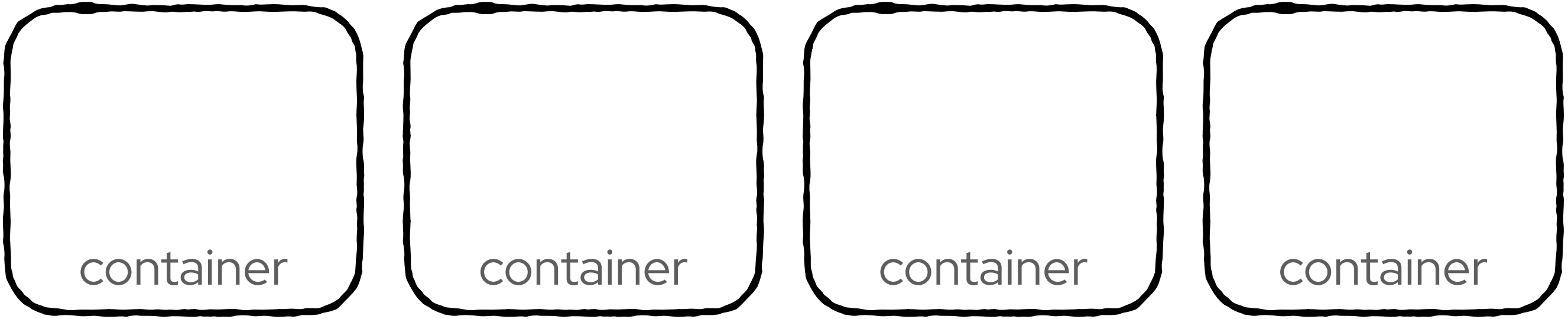
cloud way



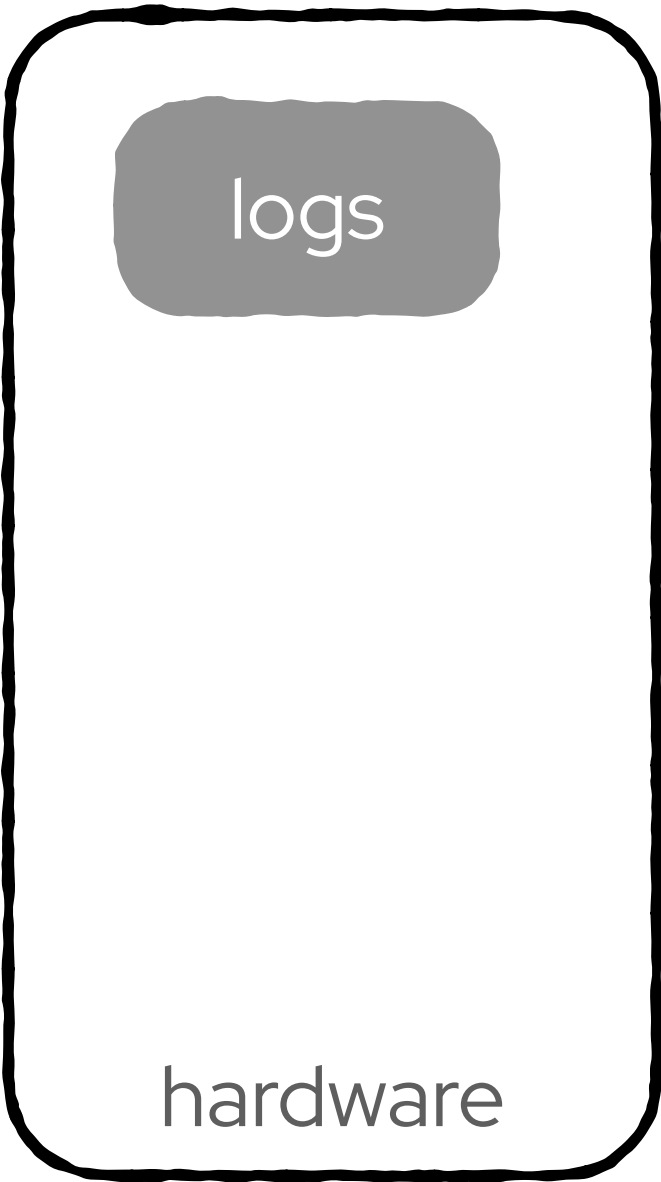
old way



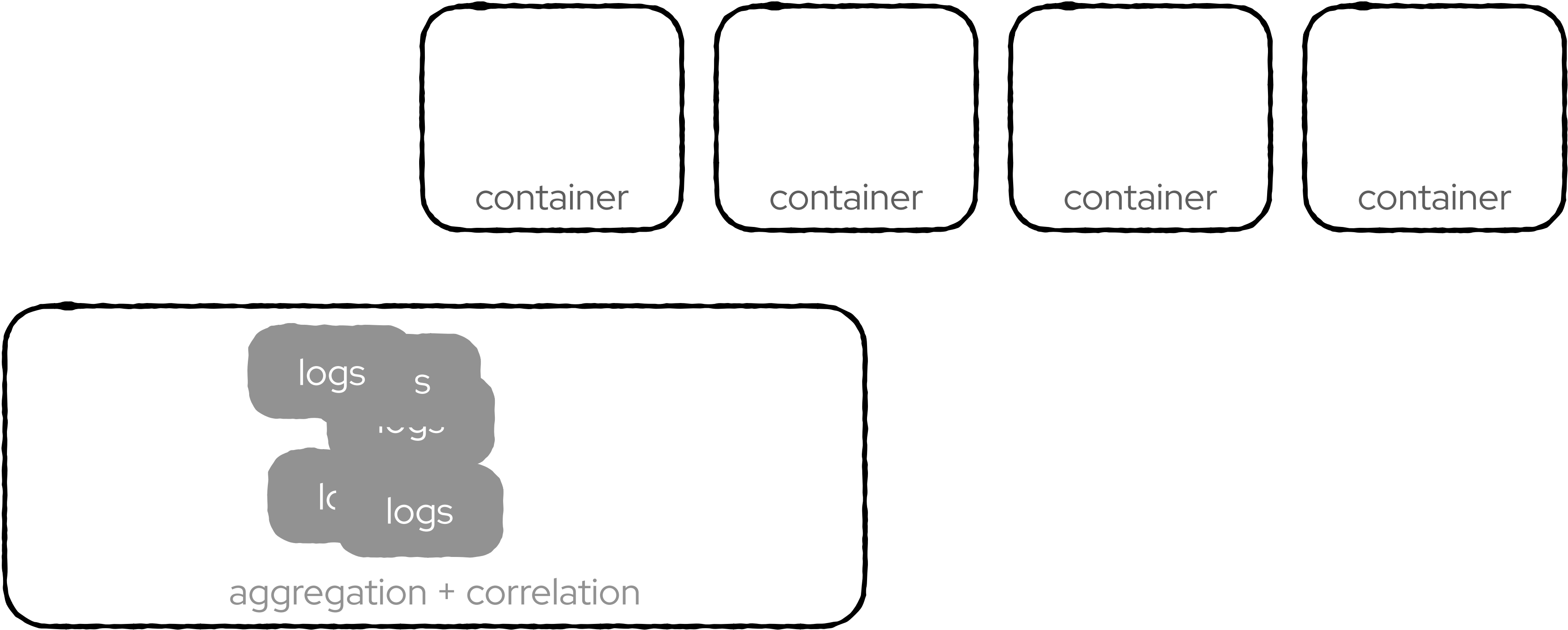
cloud way



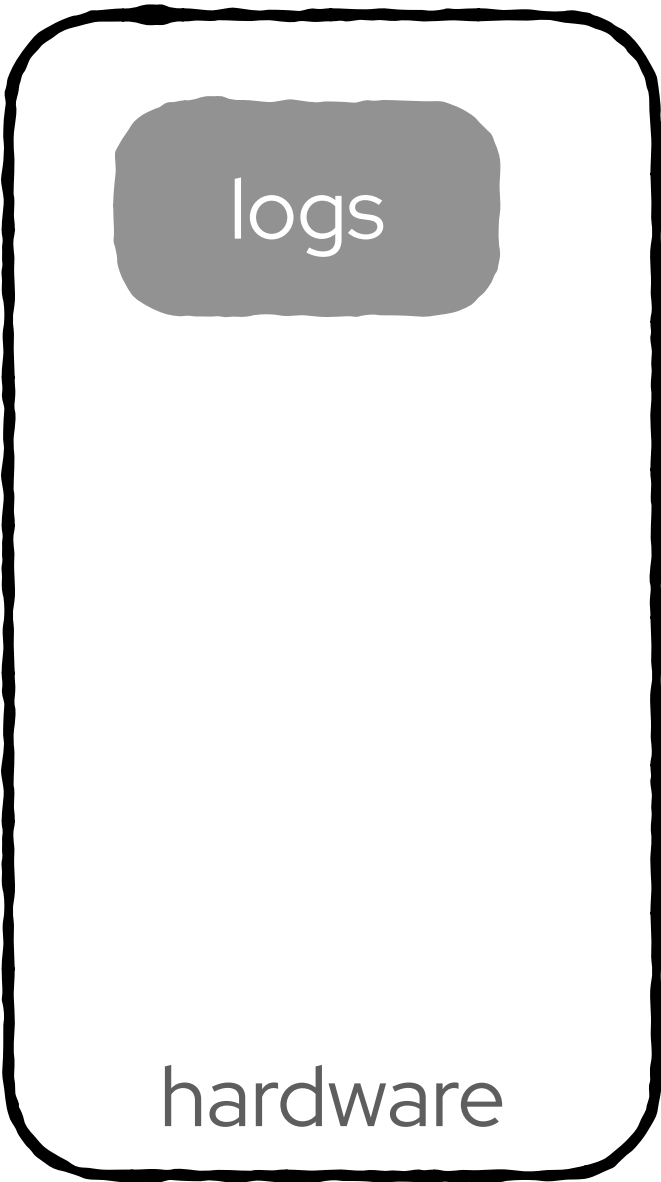
old way



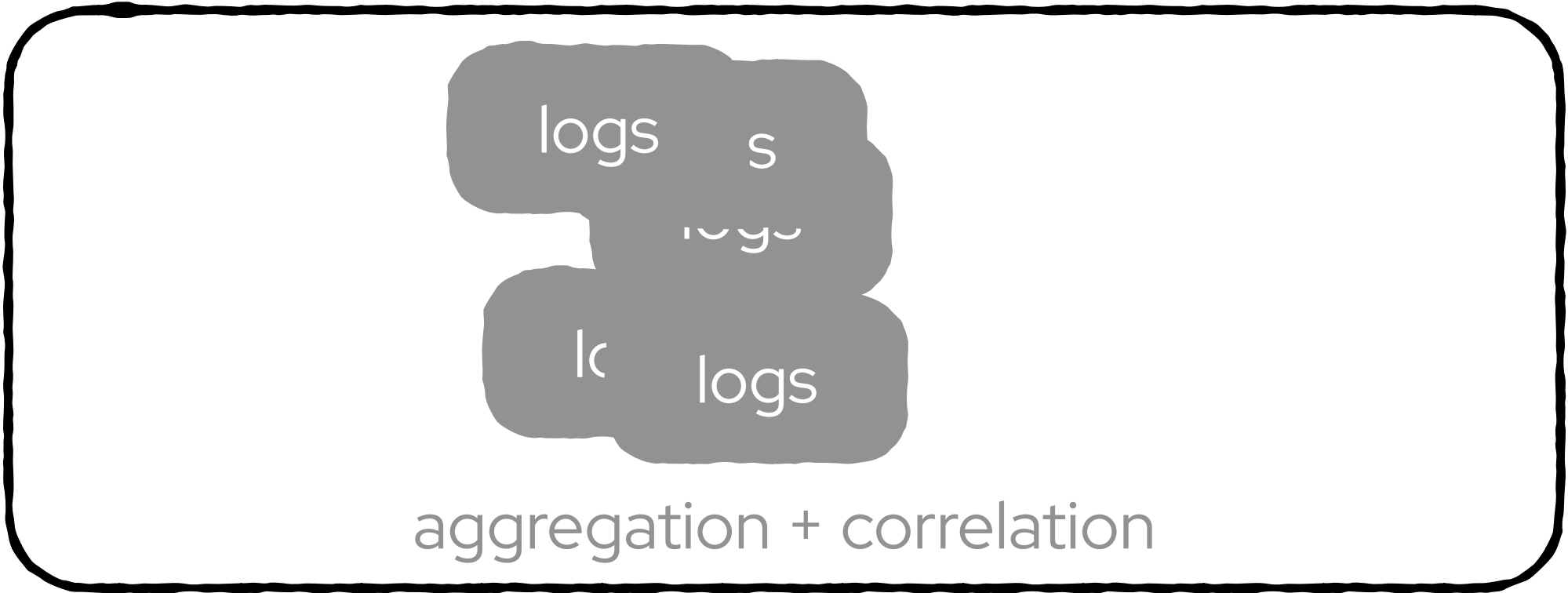
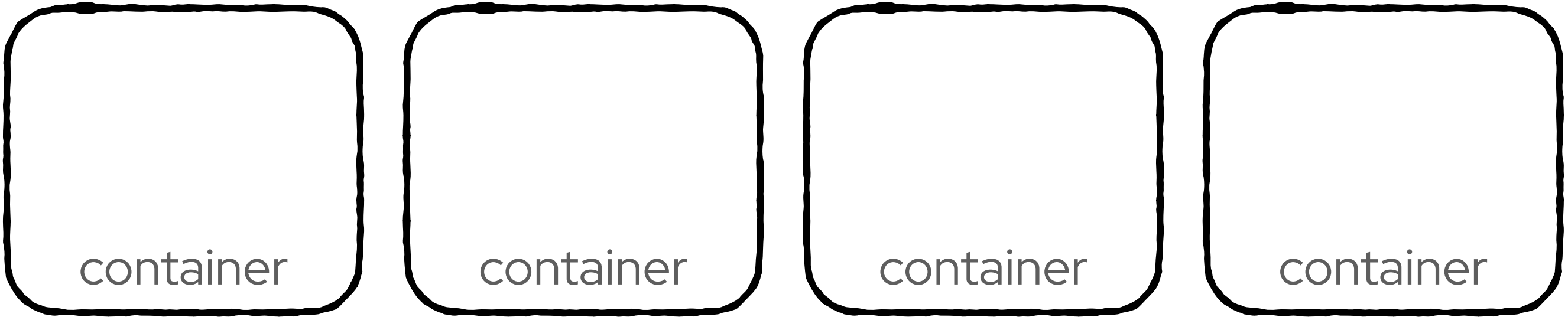
cloud way



old way



cloud way




open telemetry
(open tracing)

good news: observability is 'fixed'

good news: observability is 'fixed'
bad news: not all of us are using it

things that are
different in the



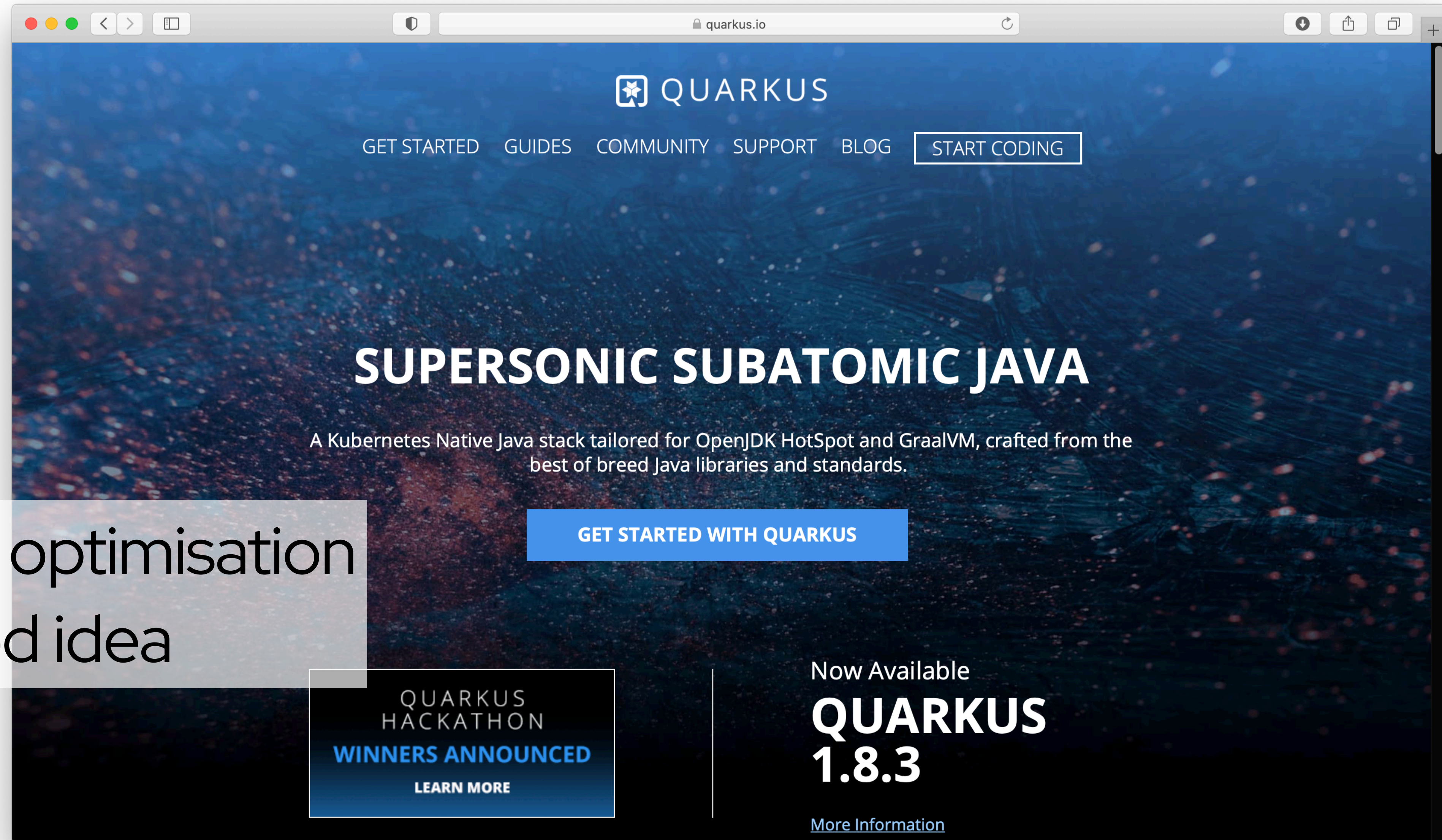
performance requirements




memory footprint is money
startup time is money

all that dynamism we worked so
hard is (now) a bad idea

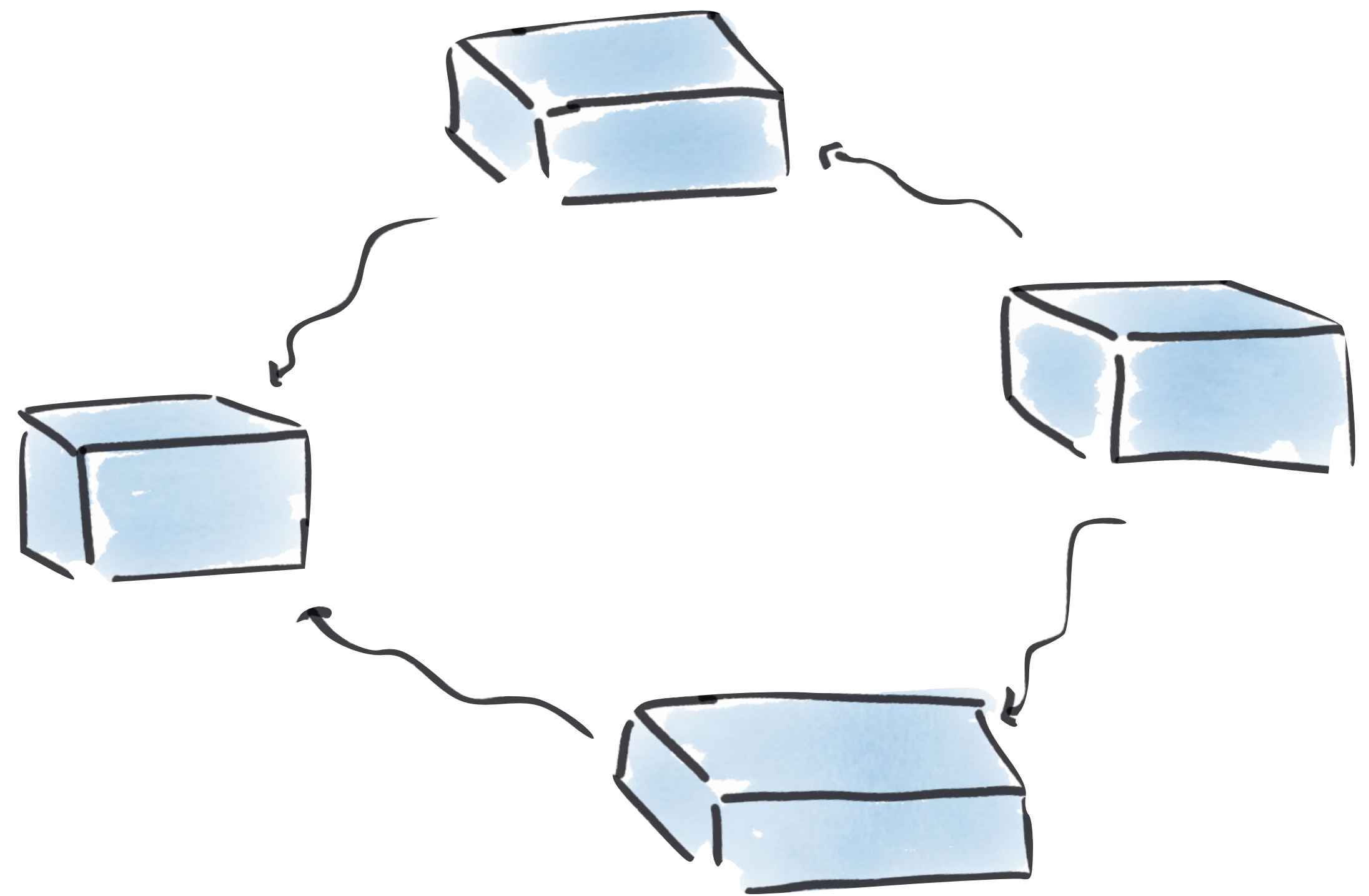
ahead of time optimisation
is (now) a good idea



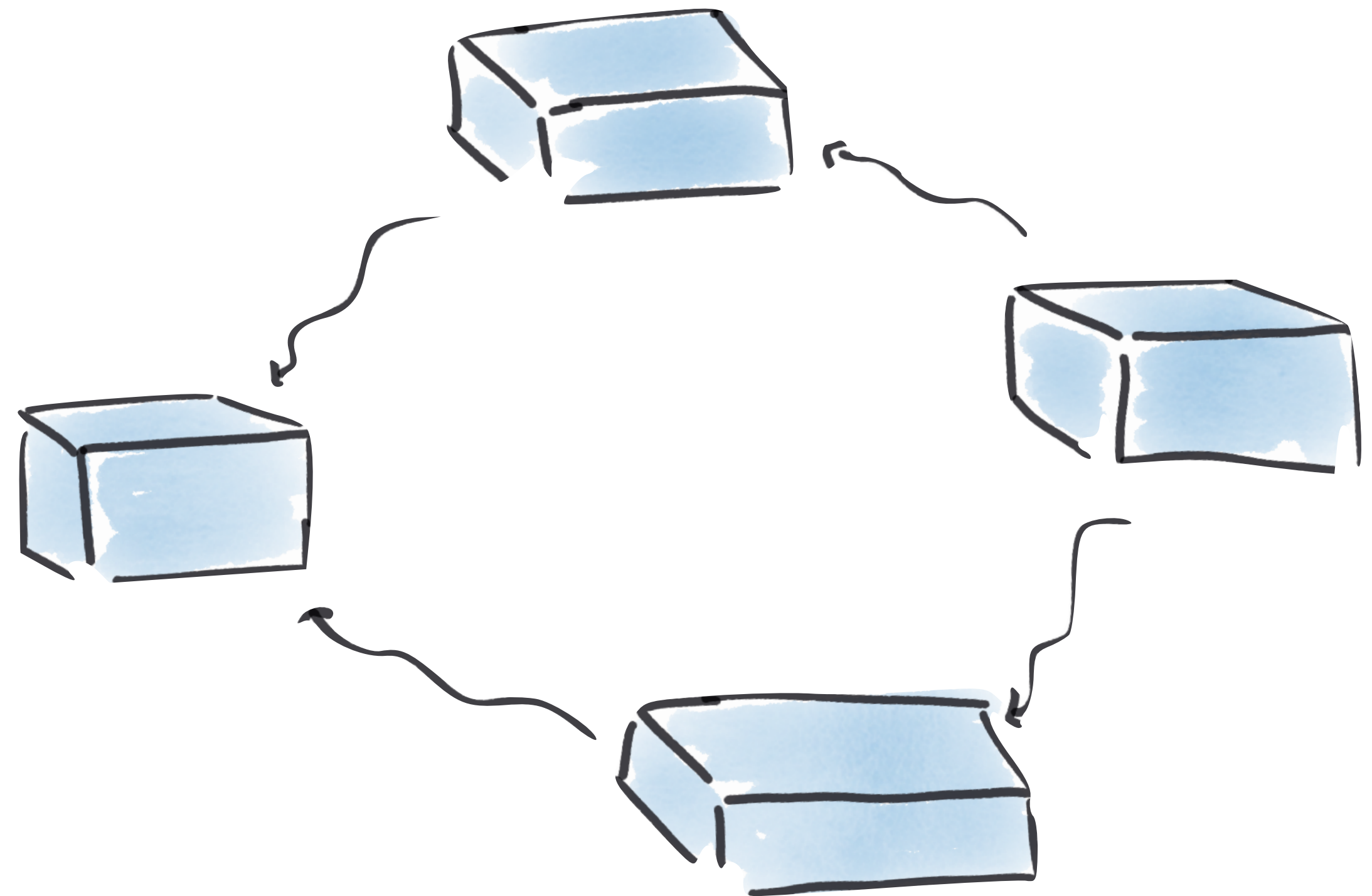
things that are
different in the



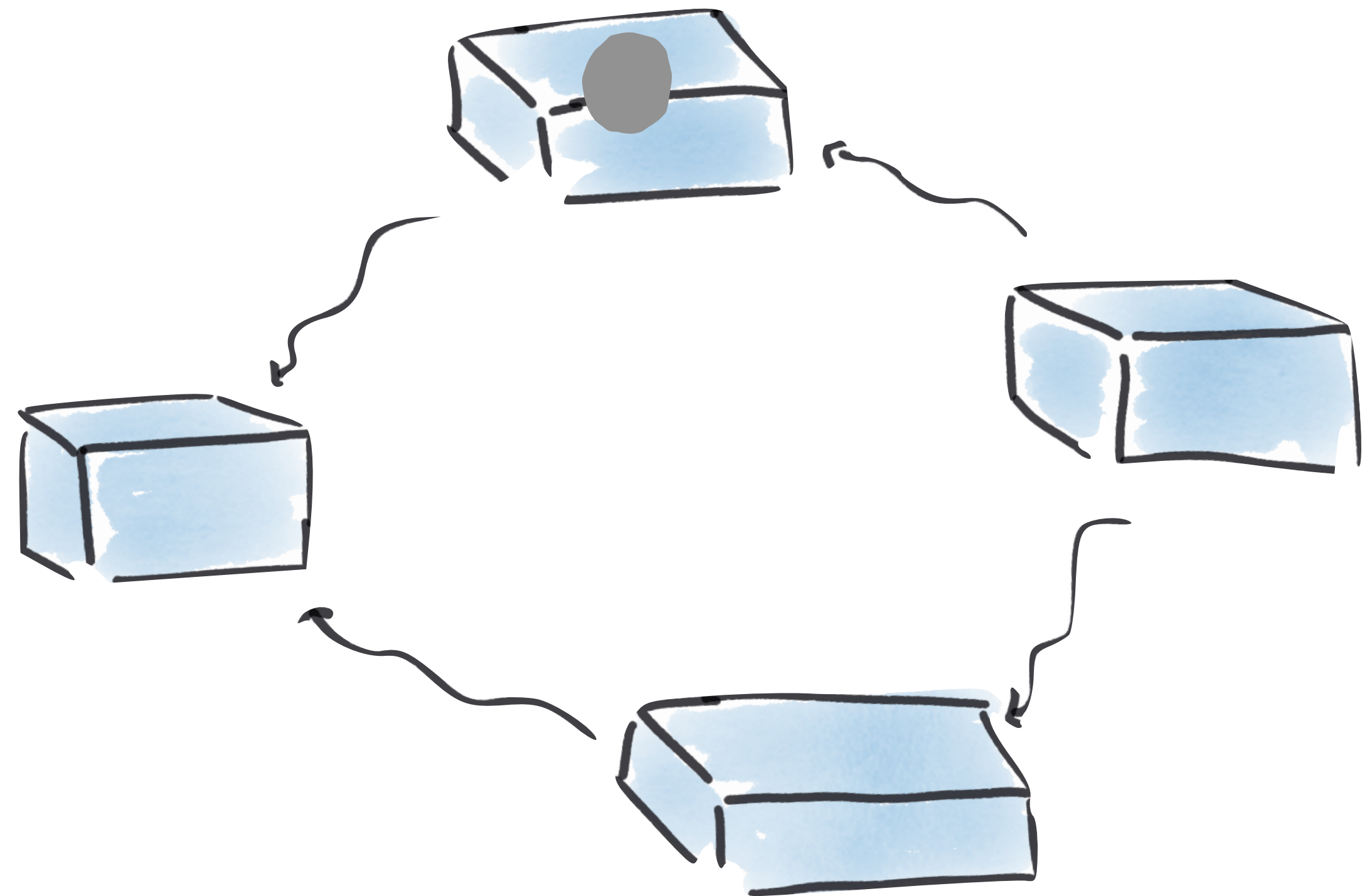
performance optimisation



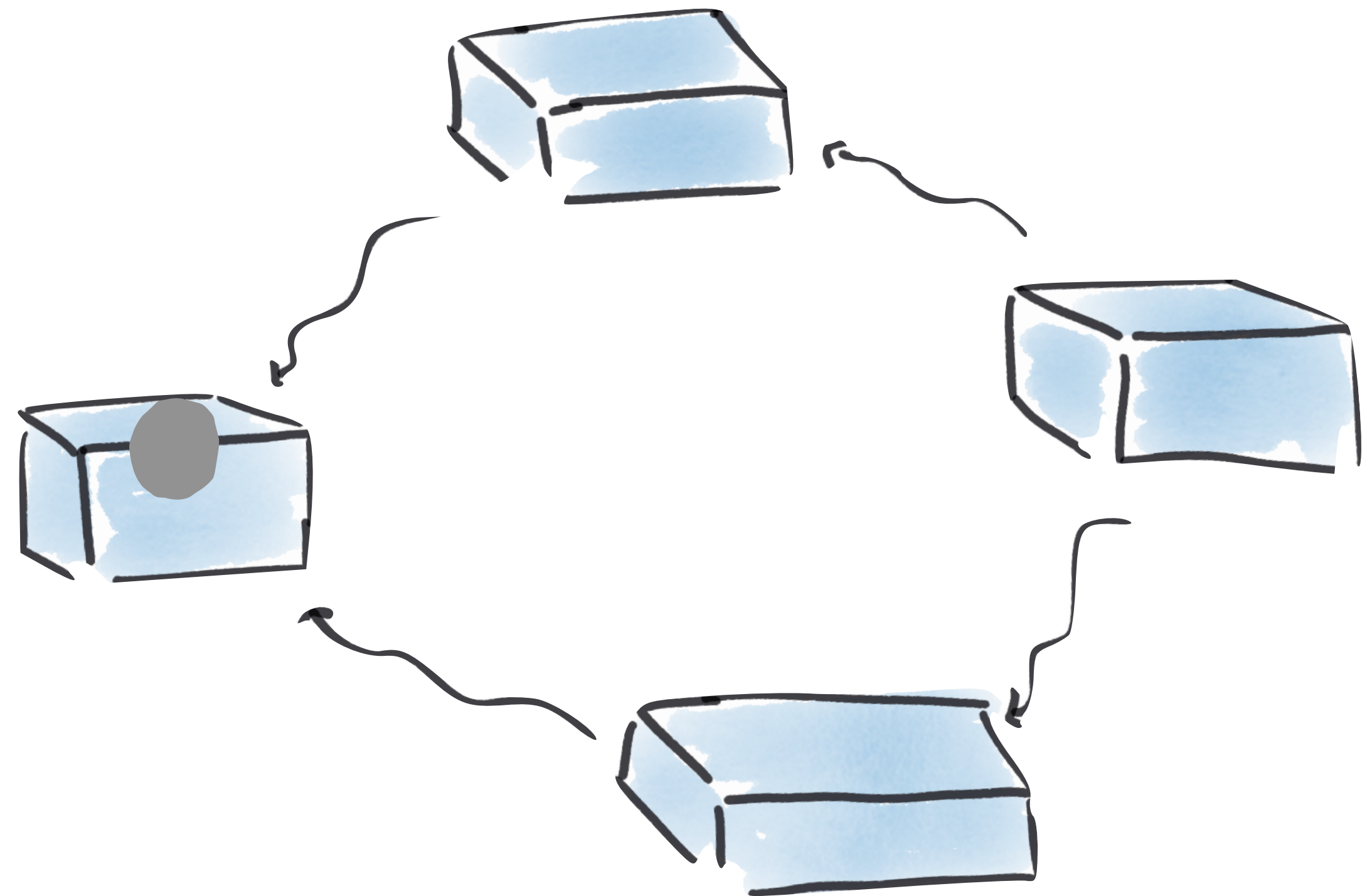
garbage collection and microservices



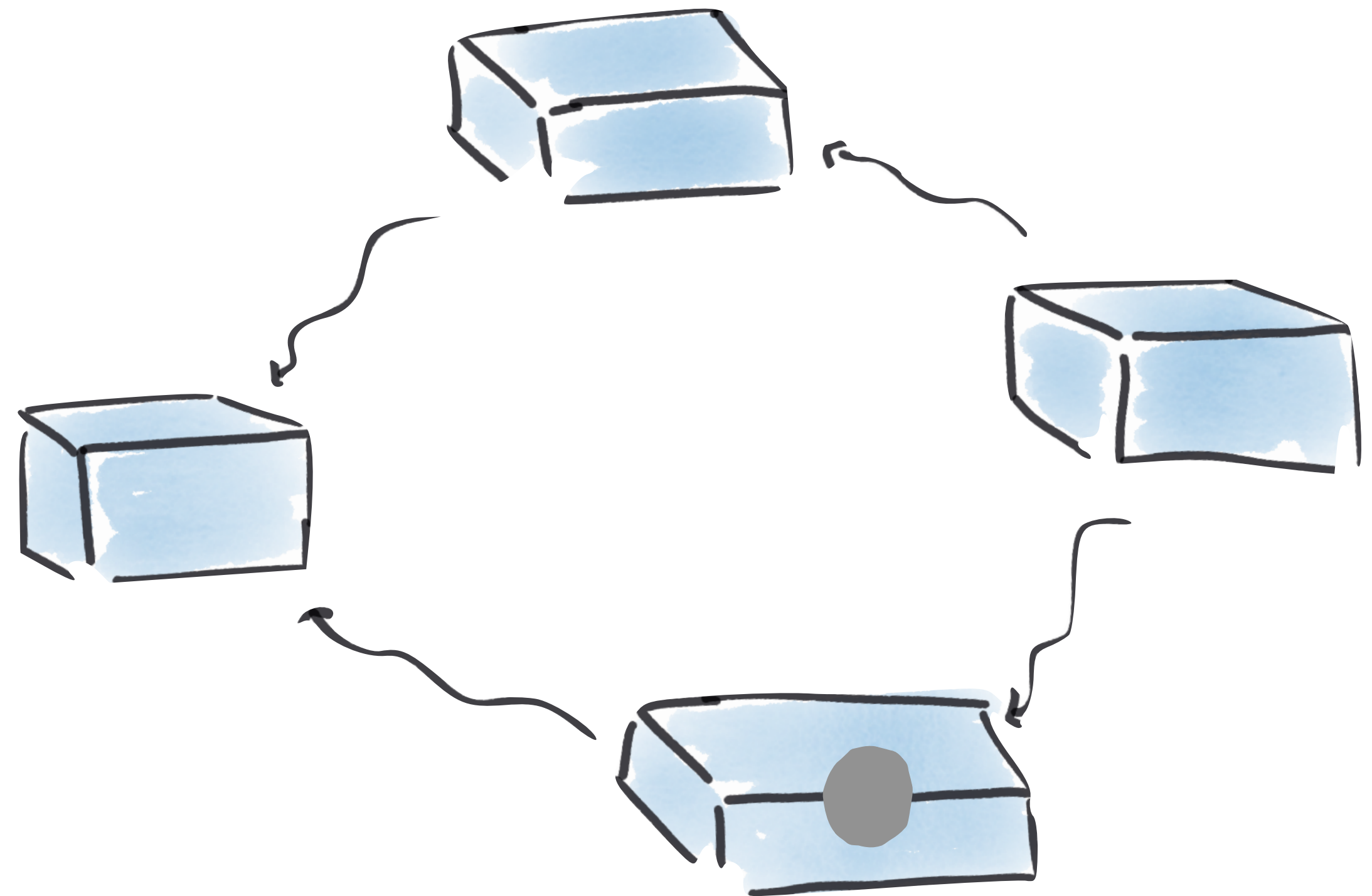
garbage collection and microservices



garbage collection and microservices

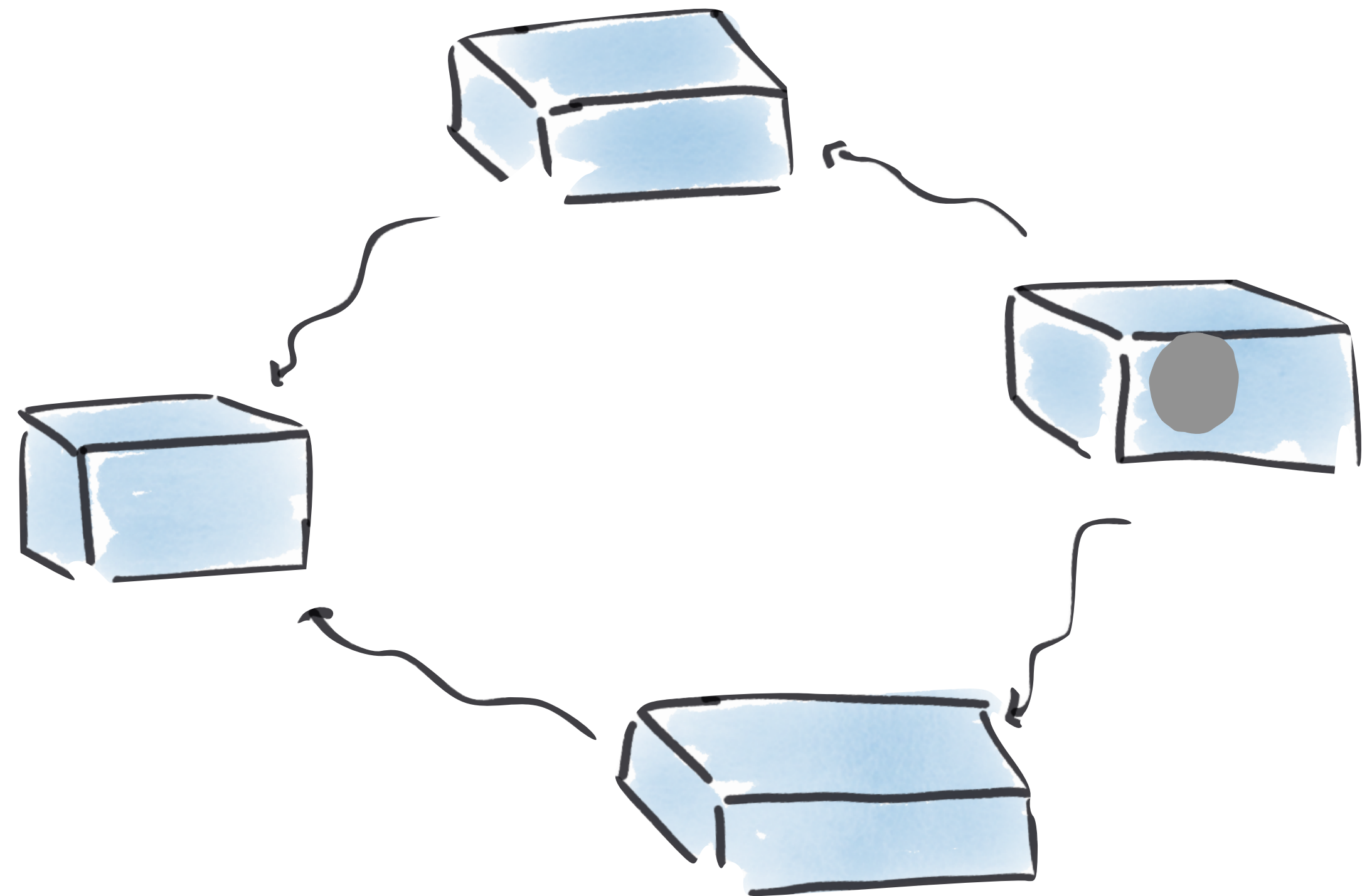


garbage collection and microservices



garbage collection and microservices

if you chain 100 microservices together,
you are almost guaranteed a GC pause



Best practices for Java in single

+

← → ↺ 🏠

🔒 https://developers.redhat.com/articles/2022/04/19/best-practices-java-single-core-containers

📄 ☆

📧 ⬇️ me ☰

🌐 Rover Apps

🔴 The Source

🌐 IT Toolbox

🔴 IT New Hire Hub

🌐 Support

📁 Presentation Resour...

📁 Red Hat External

🔴 Quarkus - Compone...

🔴 Community Spotligh...

Red Hat

Developer

Topics

Products

Developer Sandbox

Build

Tools

Events

Learn

Partner

🔍

⋮

👤

Search

All Red Hat

Log in

Article

Best practices for Java in single-core containers

April 19, 2022

🐦


f

in

✉️

🏷️

Containers, Java



Ben Evans

Senior Principal Software Engineer

📄 Table of contents:


▼

An increasing number of [Java](#) applications run in [containers](#). The exact number is hard to determine, because adoption of containers depends upon the market segment and cloud maturity of each particular team or company. However, some data is available—for example, [data from New Relic](#) suggests that over 62% of their customers' Java workloads run in containers. Like all data points, this one is an imperfect proxy for the market as a whole, but the report demonstrates that a significant subset of the Java market has already moved to container-based environments. Anecdotal data also tells us that this migration trend is far from over.

Teams using Java need to pay special attention to some aspects of container-based deployments and adopt a couple of best practices. This article focuses on the choice of garbage collector (GC) and how the default choice is based on available CPUs and memory.

Java application lifecycle

The traditional Java application lifecycle consists of a number of phases: Bootstrap, intense class loading, and a warmup with just-in-time (JIT) compilation, followed by a long-lived steady state lasting for days or weeks with relatively little class loading or JIT. This makes



Recent Articles

4 tips for achieving better security on Kubernetes

Use OpenVINO to convert speech to text

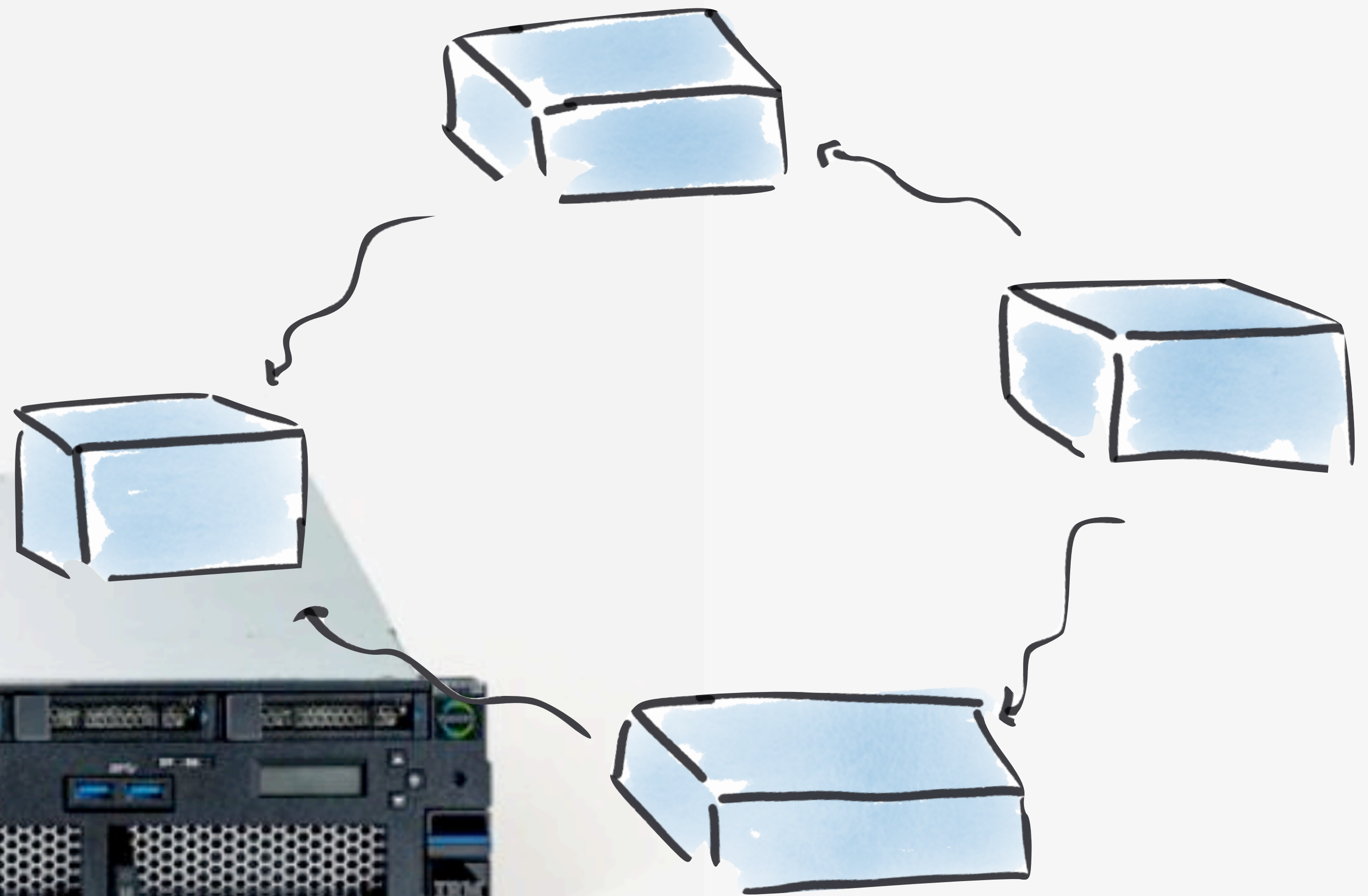
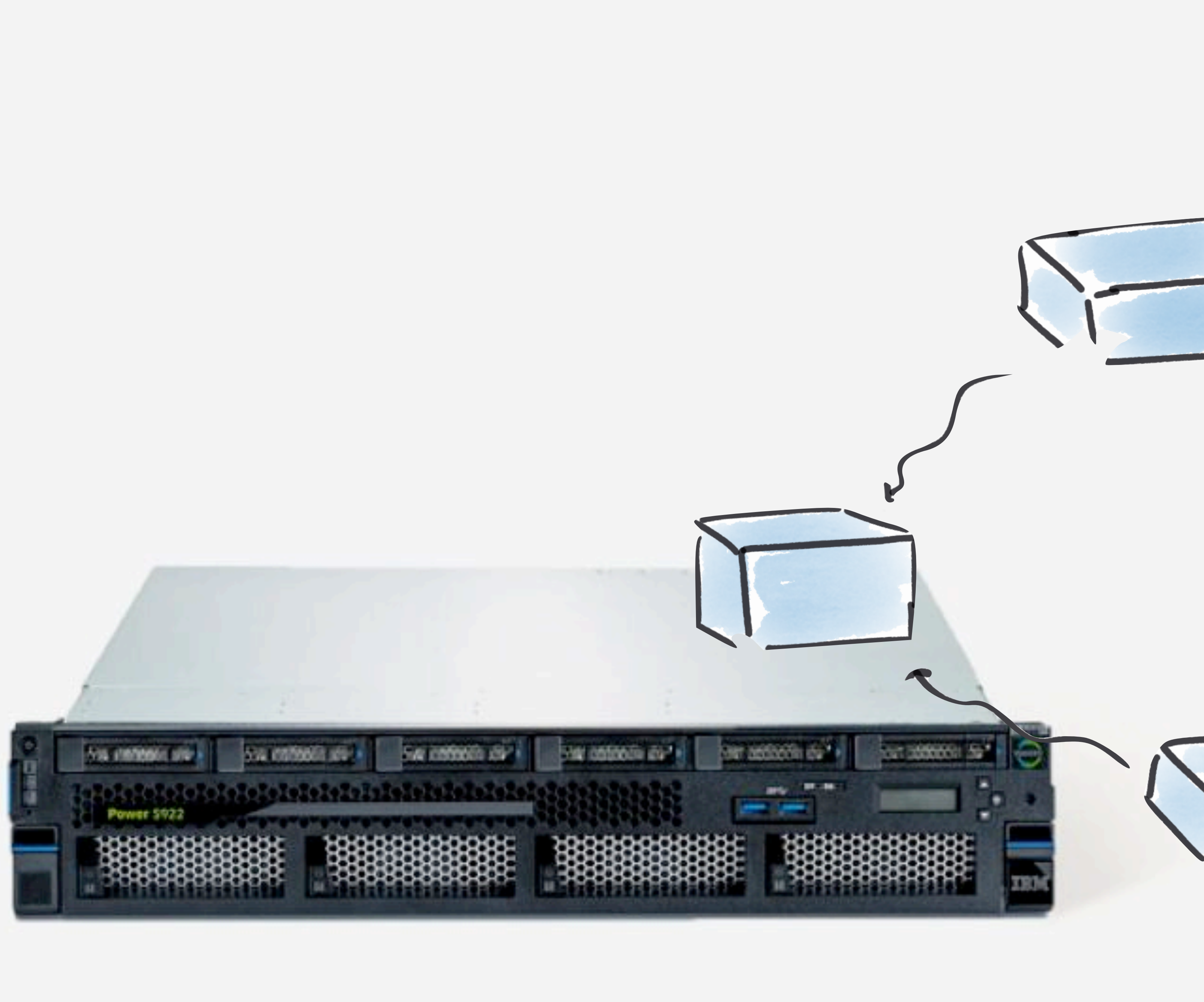
Kubernetes security risks that keep developers up at night

A quick way to translate physical addresses into virtual ones

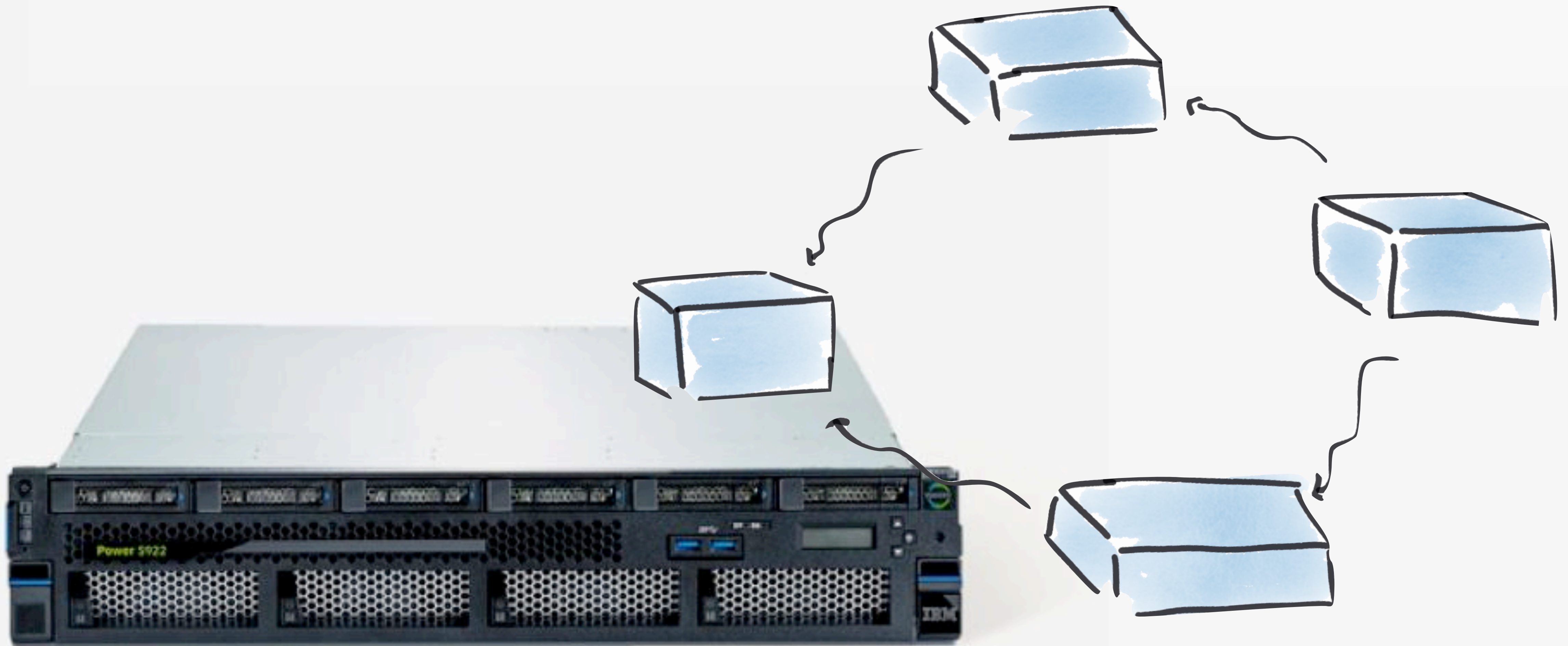
Get started with Red Hat OpenShift

@holly_cummins

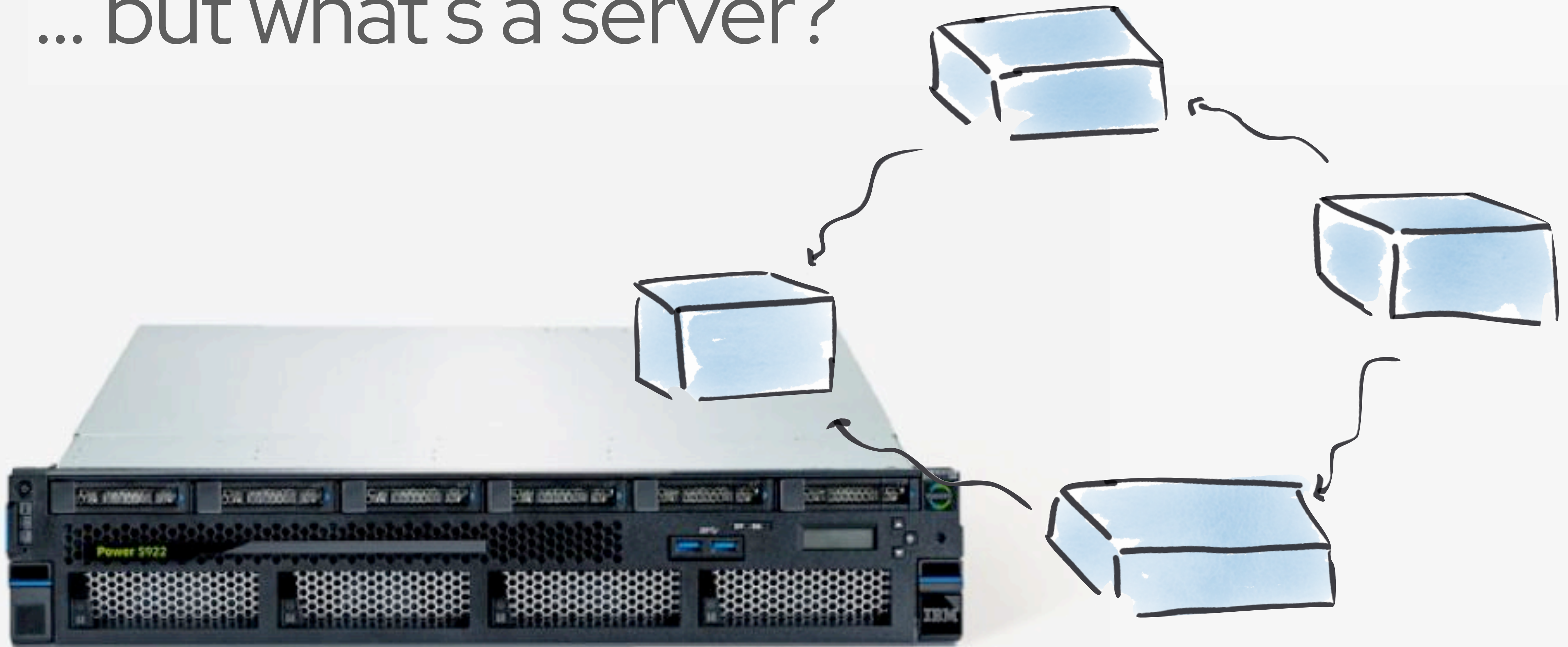
#RedHat



the JVM tunes itself differently on servers



the JVM tunes itself differently on servers
... but what's a server?



answer(ish):



answer(ish):

```
// This is the working definition of a server class machine:  
//  $\geq 2$  physical CPU's and  $\geq 2$ GB of memory
```



actual hotspot source code

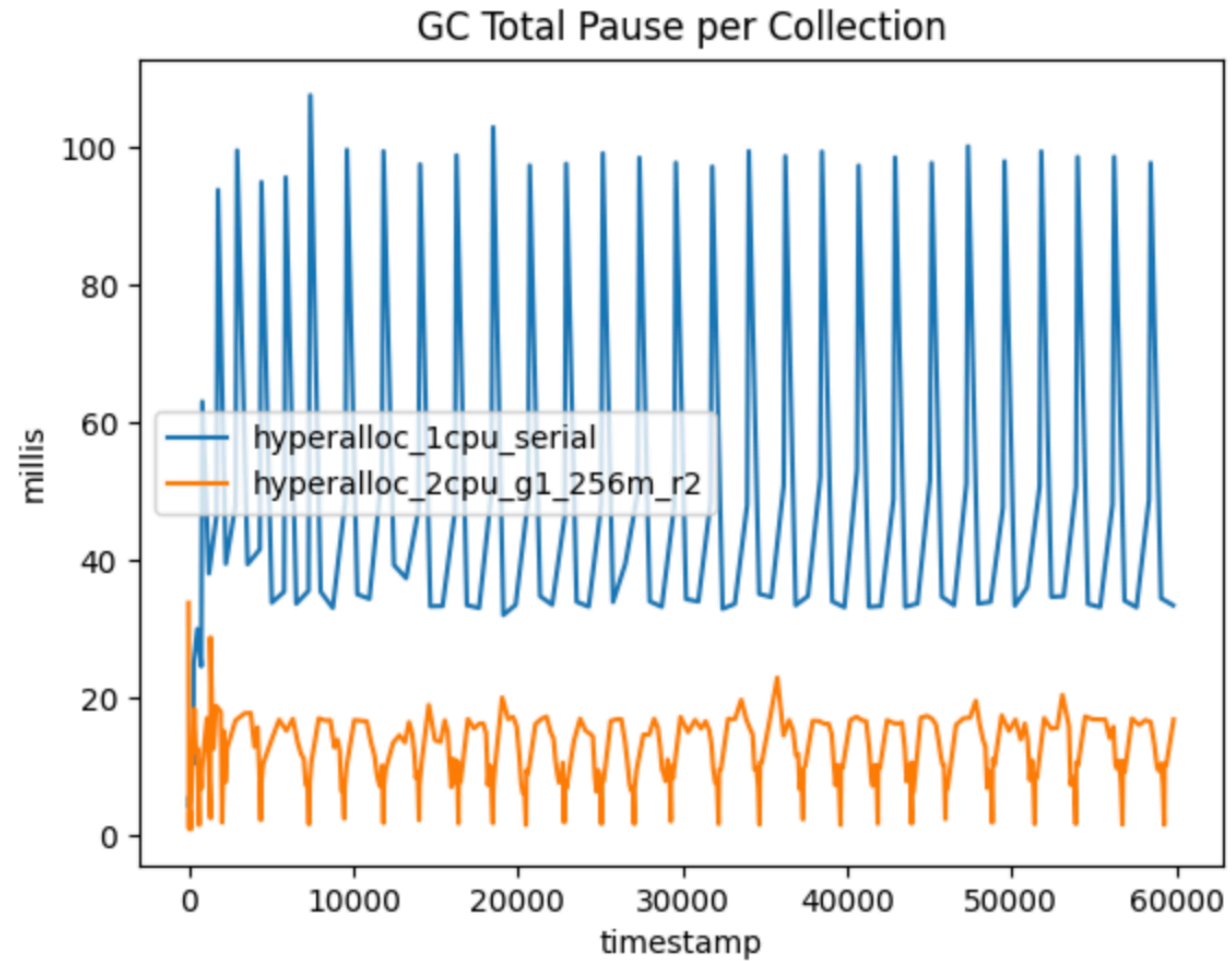
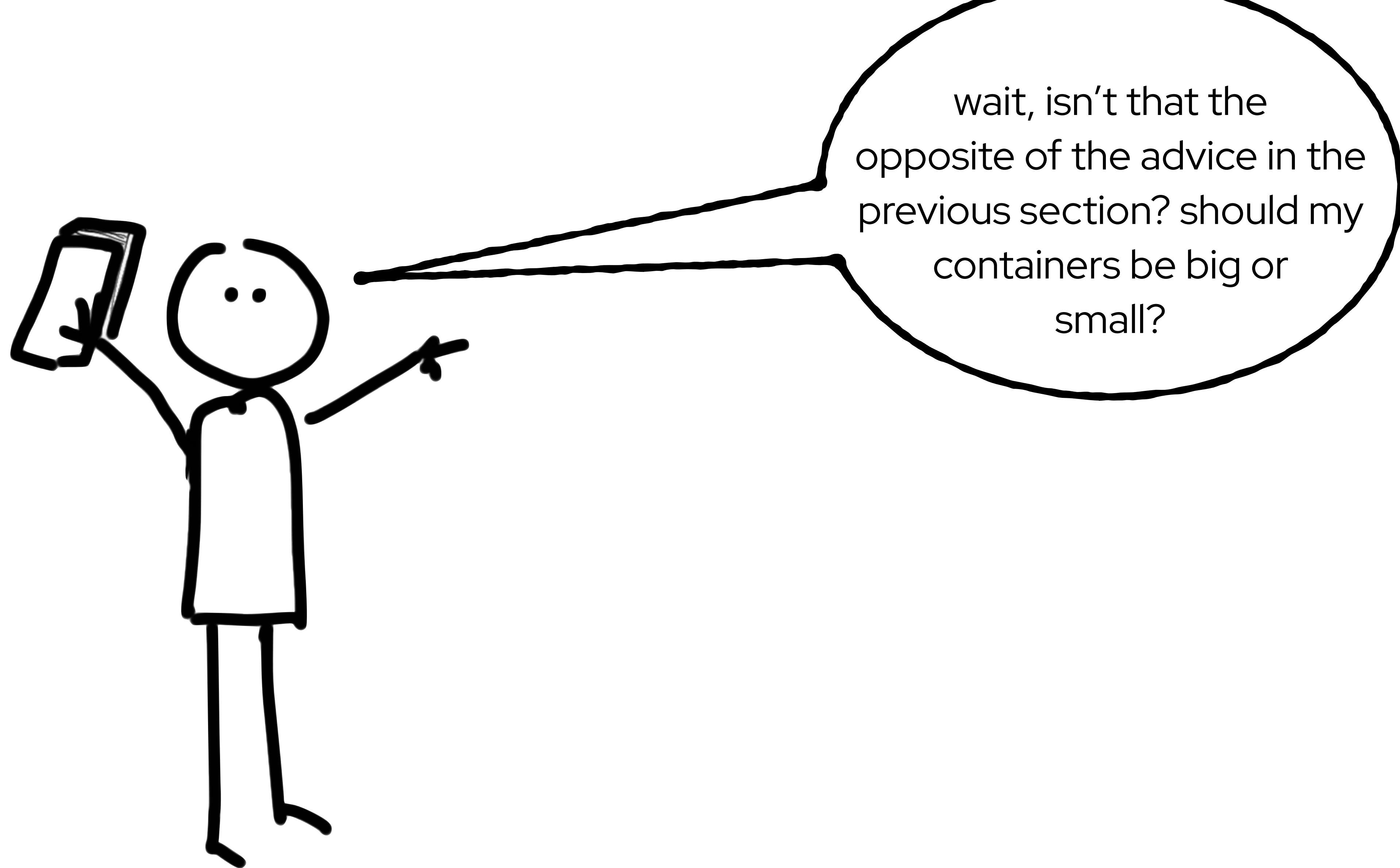


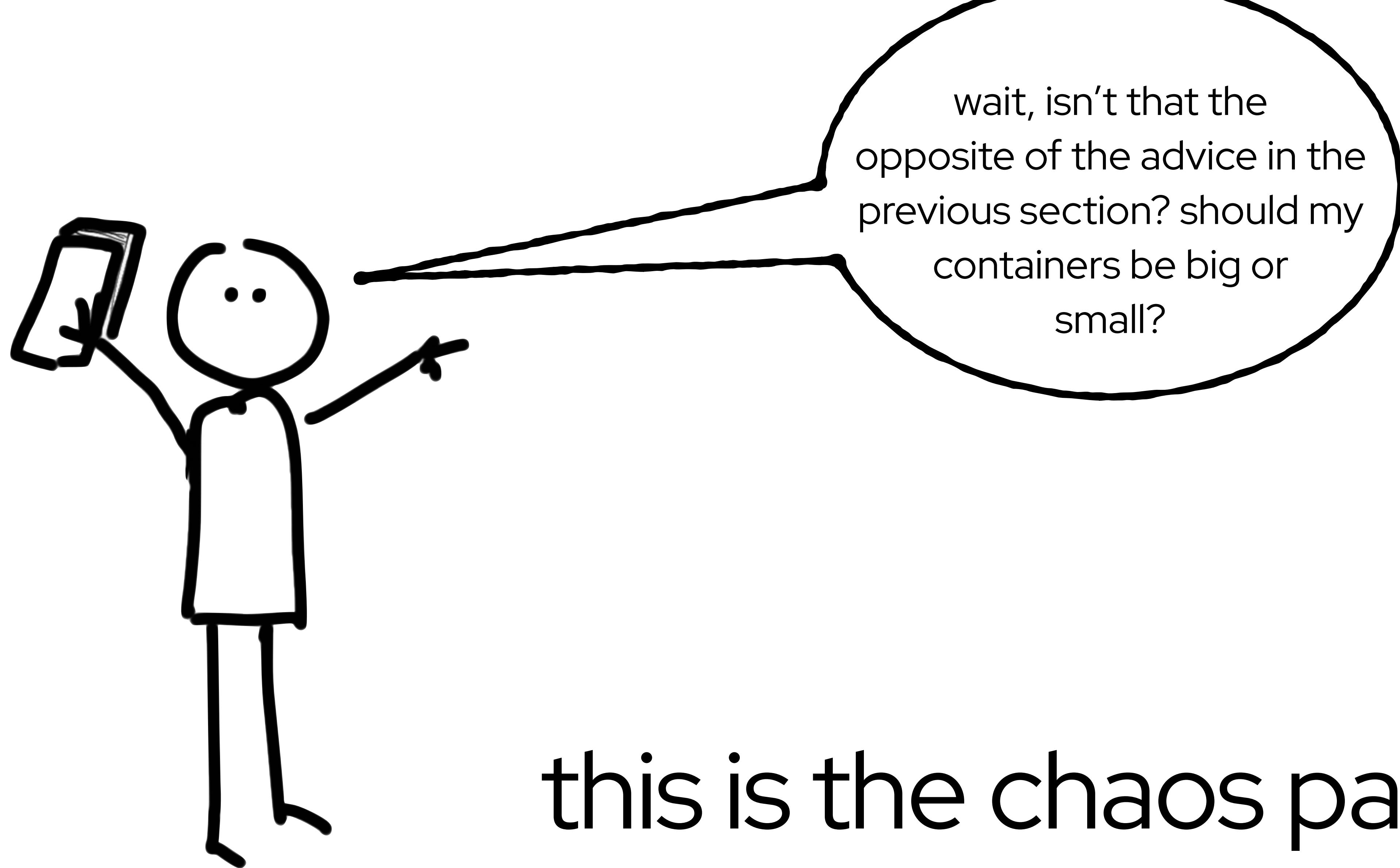
Figure 1: The one-CPU run has much higher total pauses than the two-CPU run.

takeaway: don't shrink your containers too much or your JVM will make bad decisions

advice accurate as of June 2022. all performance advice should be independently verified. do not try this at home without testing first. your mileage may vary.




wait, isn't that the
opposite of the advice in the
previous section? should my
containers be big or
small?




this is the chaos part :)

things that are
different in the



management

things that are
different in the



keeping track of hardware

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.

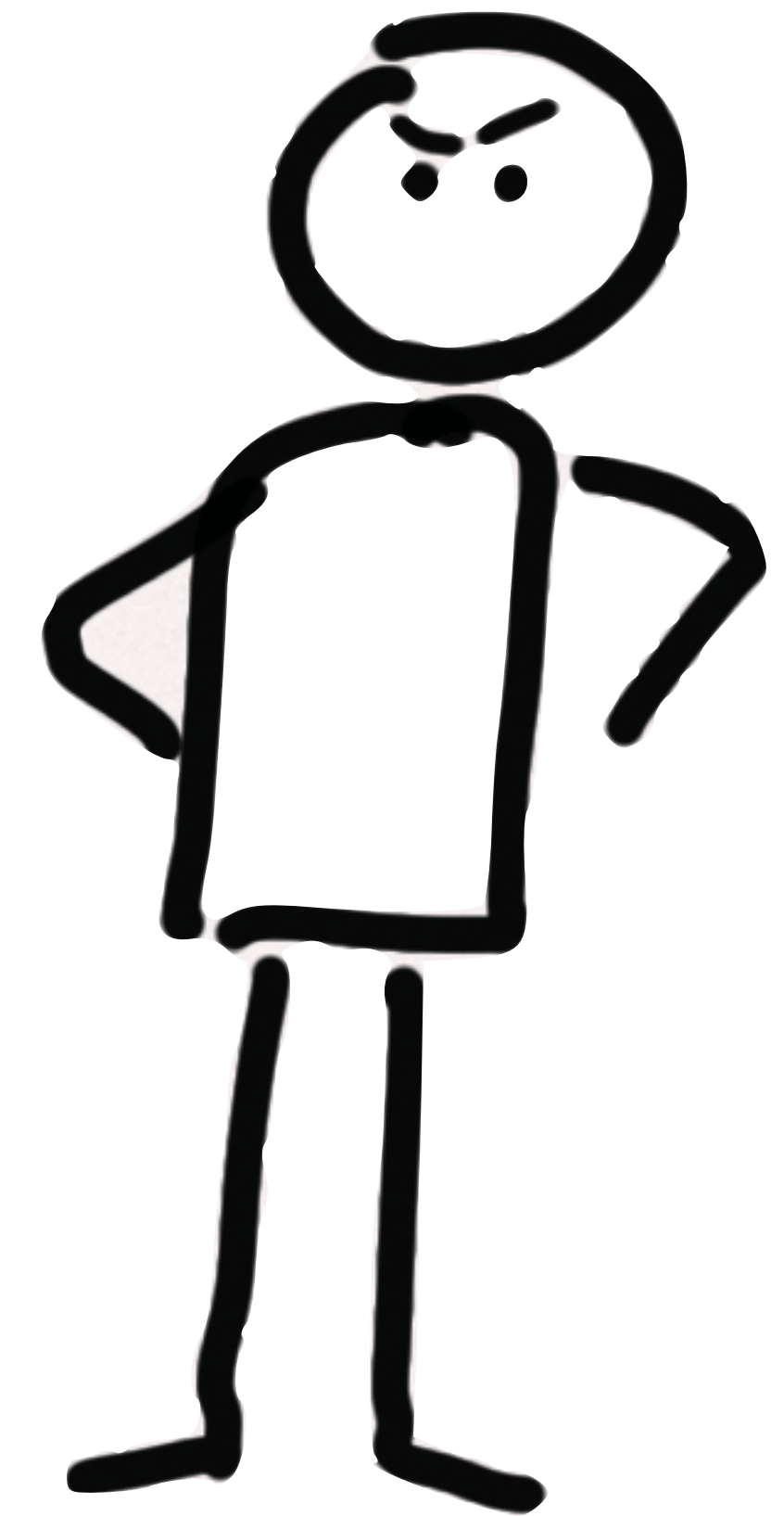
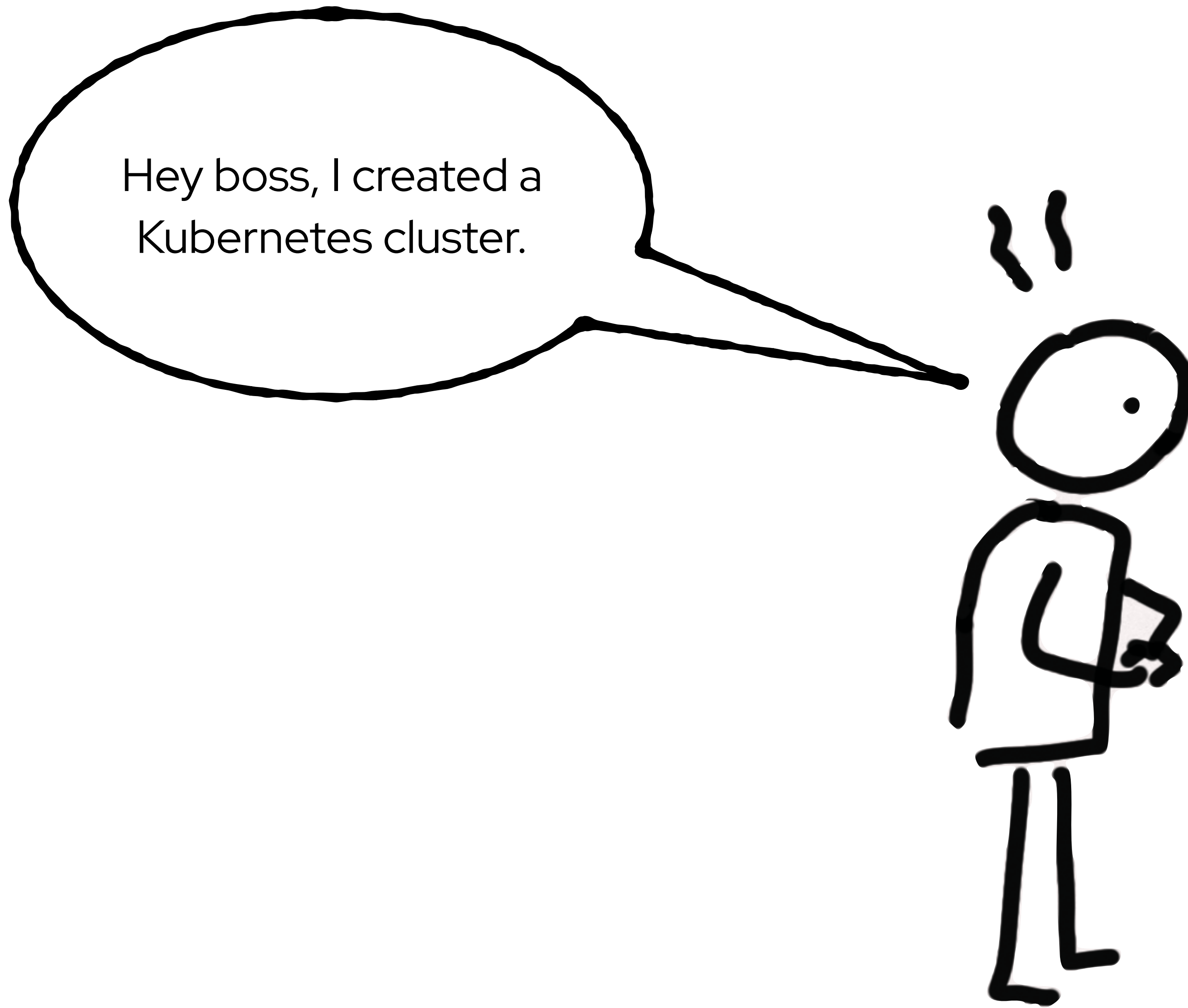
That doesn't mean the hardware is
free.

Or useful.

That doesn't mean the hardware is
free.

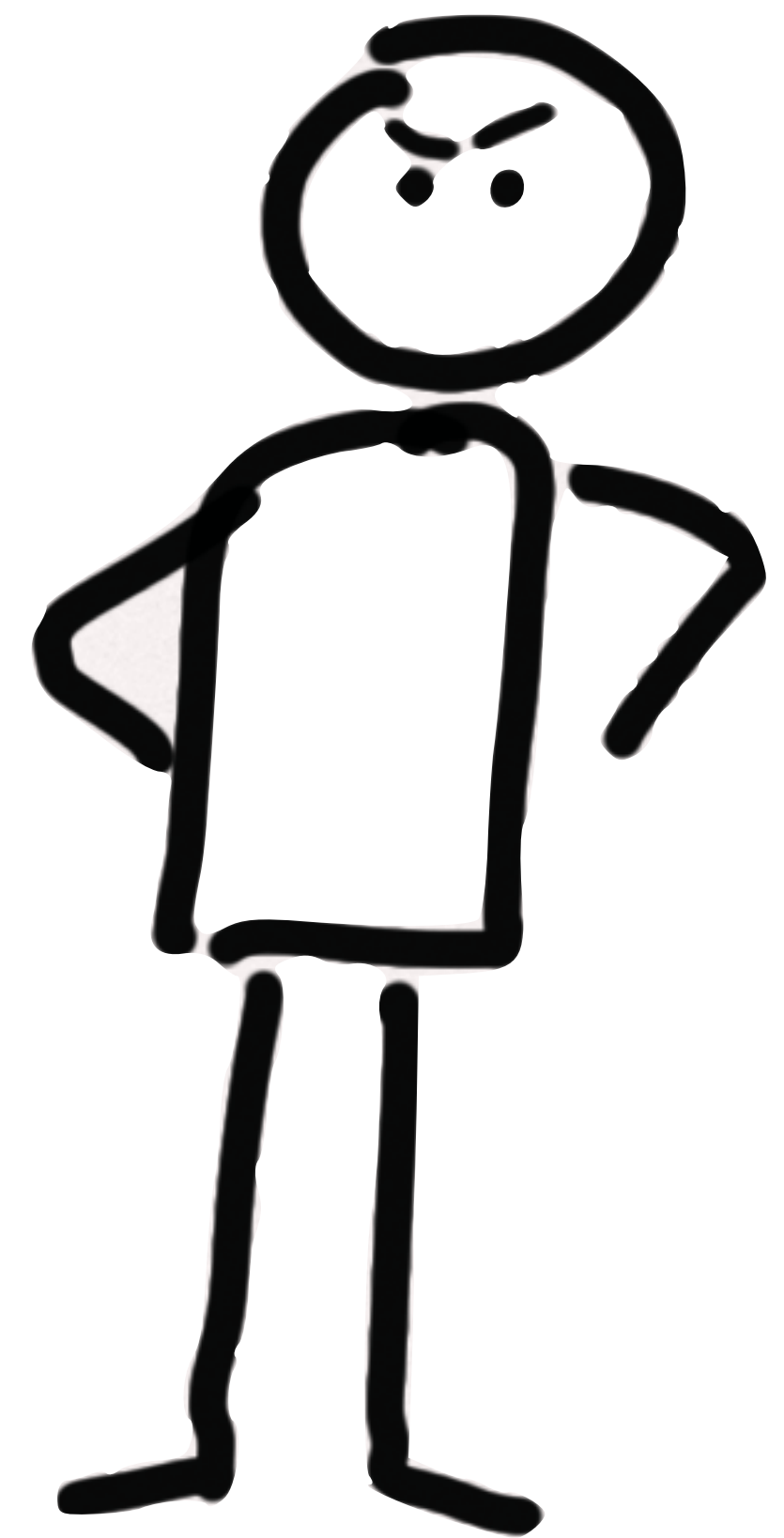
Or useful.

Or shuts itself off.



Hey boss, I created a
Kubernetes cluster.

I forgot it for 2 months.



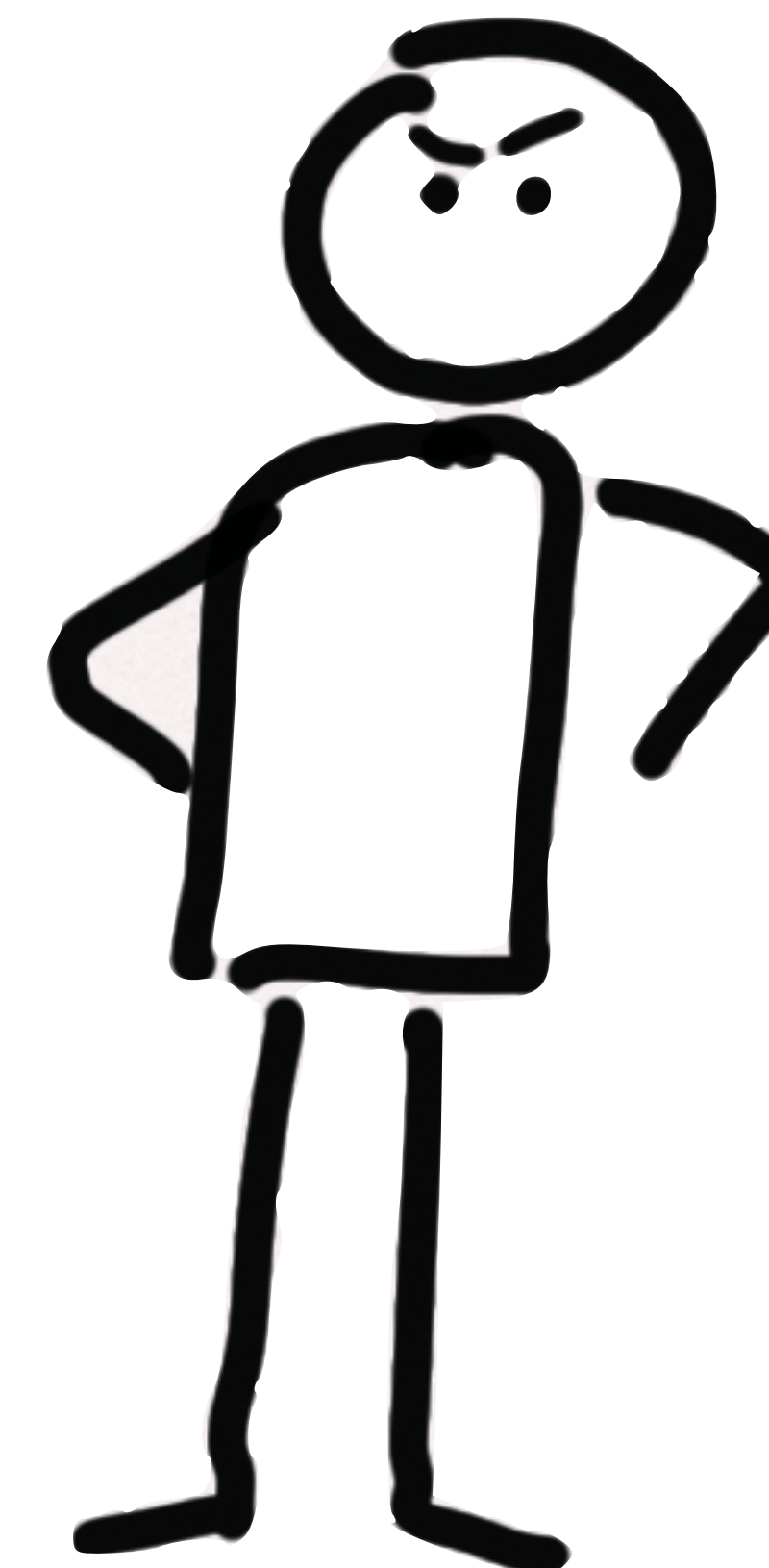
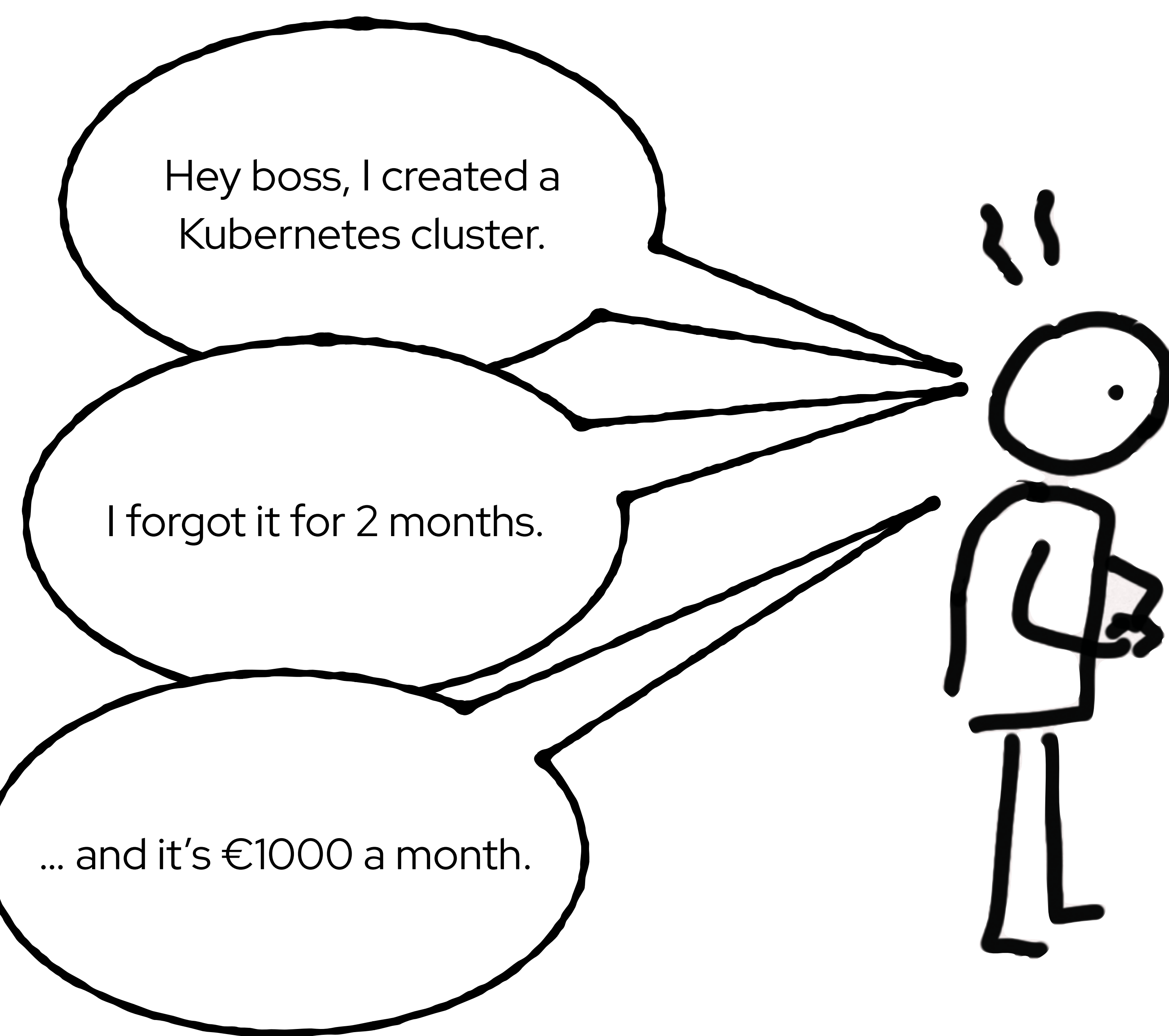




Photo by daveynin, flickr

2017 survey

25%

of 16,000 servers
doing **no** useful work



2017 survey

25%

of 16,000 servers
doing **no** useful work

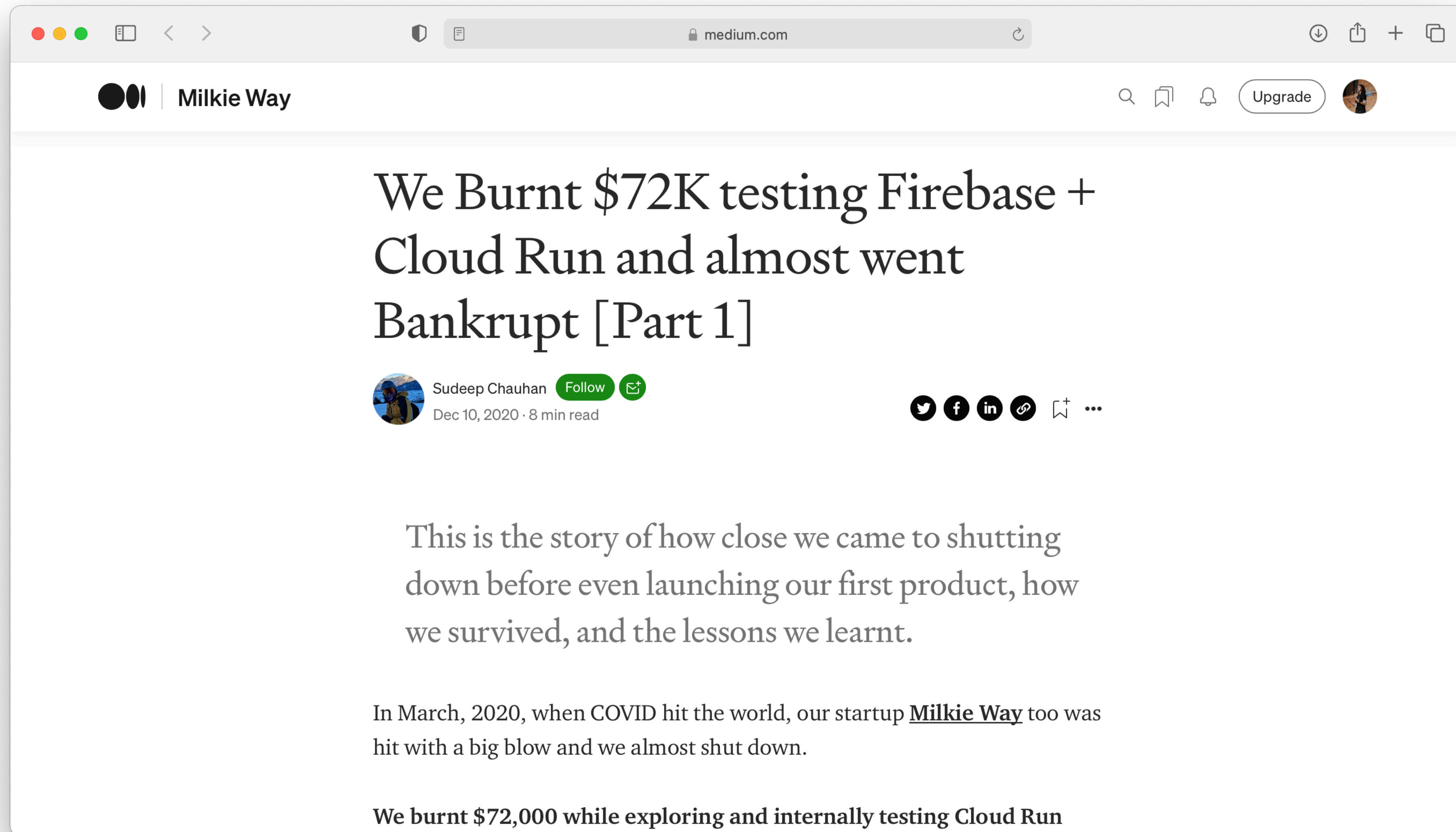
"perhaps someone
forgot to turn them off"



finops

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

companies that went out of
business for cloud cost accidents



(See Troy tomorrow at GOTO !)

troyhunt.com

HOMEWORKSHOPSPEAKINGMEDIABOUTCONTACTSPONSOR

AUS\$12K

AUS\$11.5K

AUS\$11K

AUS\$10.5K

AUS\$10K

AUS\$9.5K

AUS\$9K

AUS\$8.5K

AUS\$8K

AUS\$7.5K

AUS\$7K

AUS\$6.5K

Sponsored by: Varonis. Reduce your ransomware blast radius with the leader in data-first security. Try it free!

How I Got Pwned by My Cloud Costs

24 JANUARY 2022

I have been, and still remain, a massive proponent of "the cloud". I built [Have I Been Pwned](#) (HIBP) as a cloud-first service that took advantage of modern cloud paradigms such as Azure Table Storage to massively drive down costs at crazy levels of performance I never could have achieved before. I wrote many blog posts about

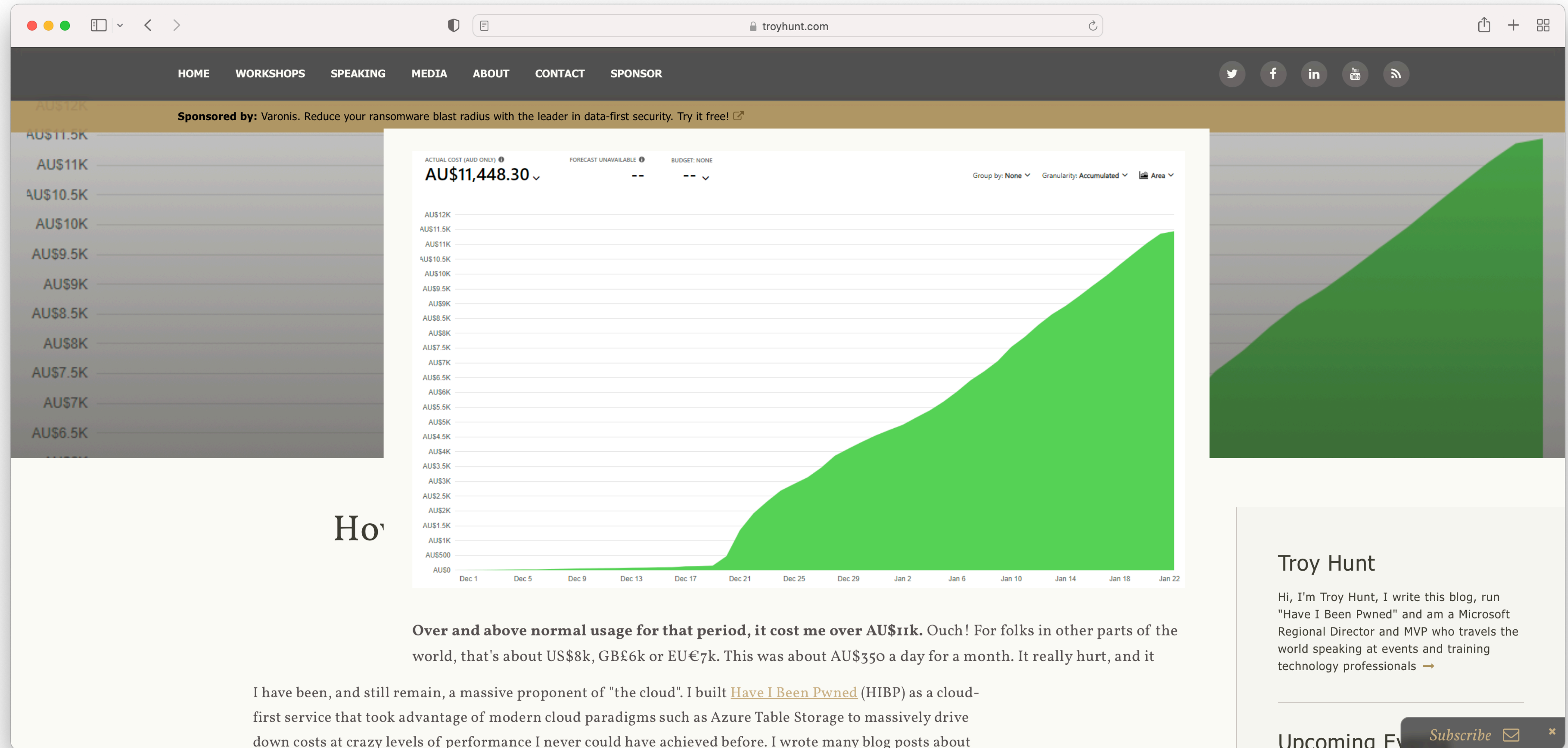
Troy Hunt

Hi, I'm Troy Hunt, I write this blog, run "Have I Been Pwned" and am a Microsoft Regional Director and MVP who travels the world speaking at events and training technology professionals →

Upcoming Events

Subscribe

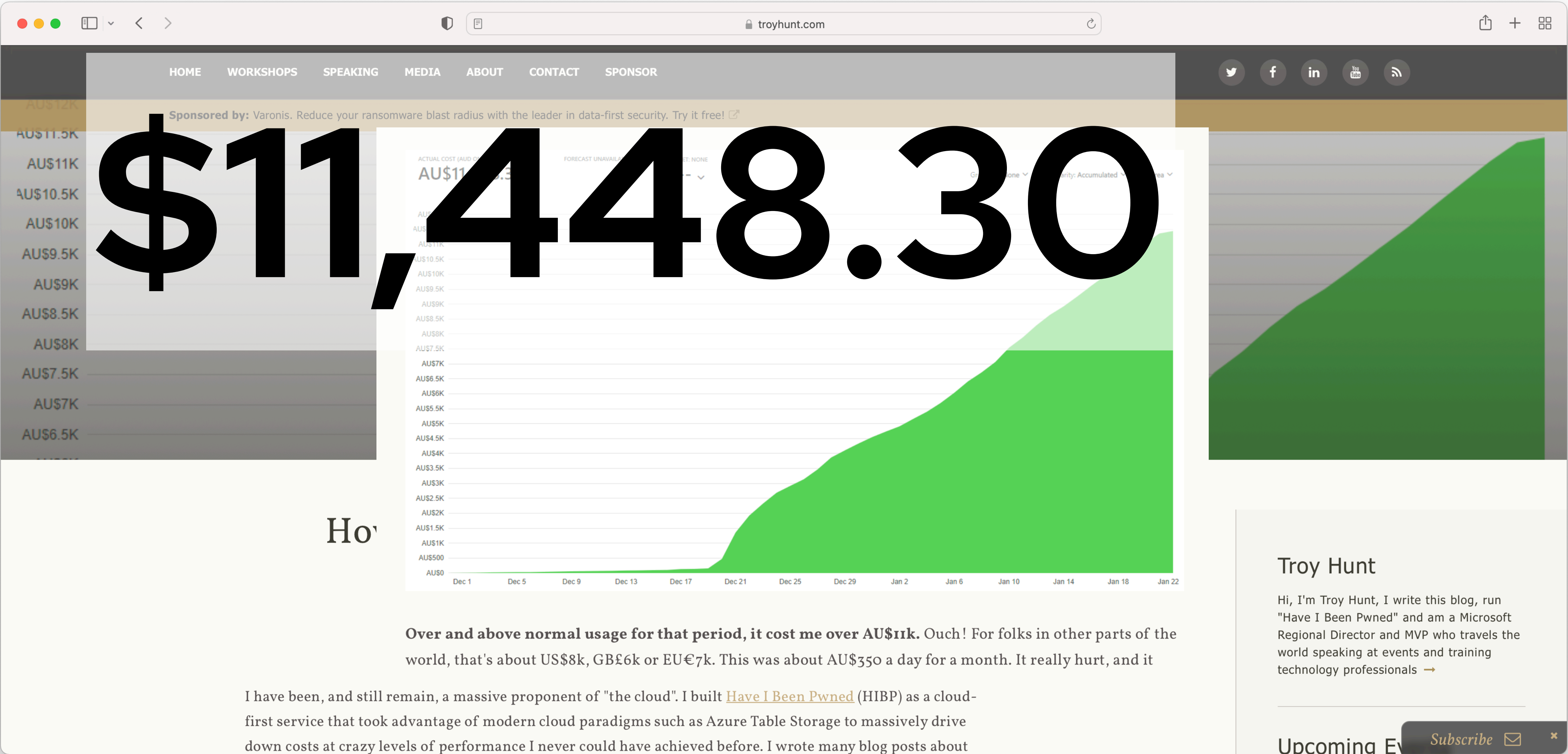
(See Troy tomorrow at GOTO !)



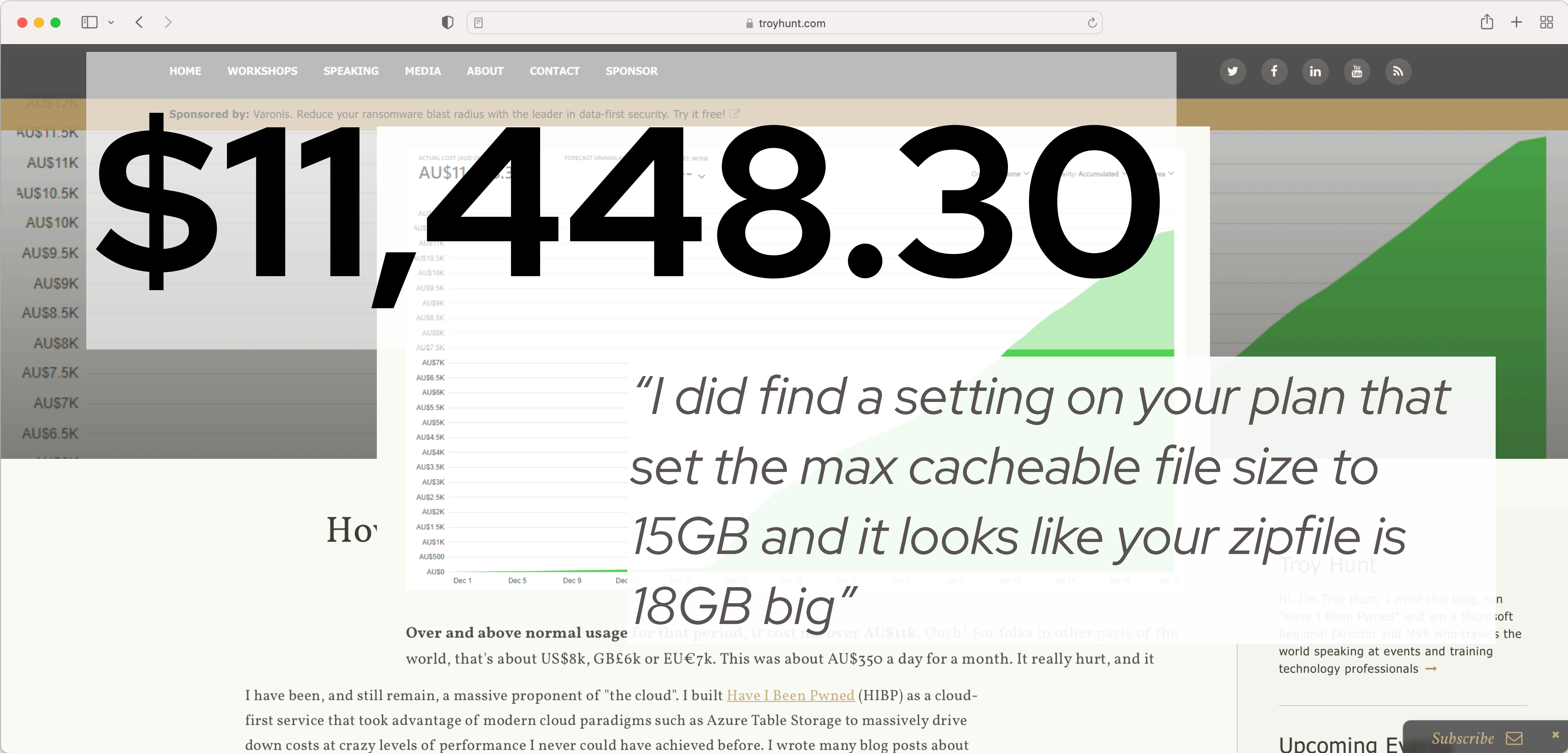
@holly_cummins

#RedHat

(See Troy tomorrow at GOTO !)



(See Troy tomorrow at GOTO !)







maybe this is the chaos part

things that are
different in the

releasing



things that are
different in the



releasing



things that are
different in the
cloud

microservices



things that are
different in the



microservices



“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**

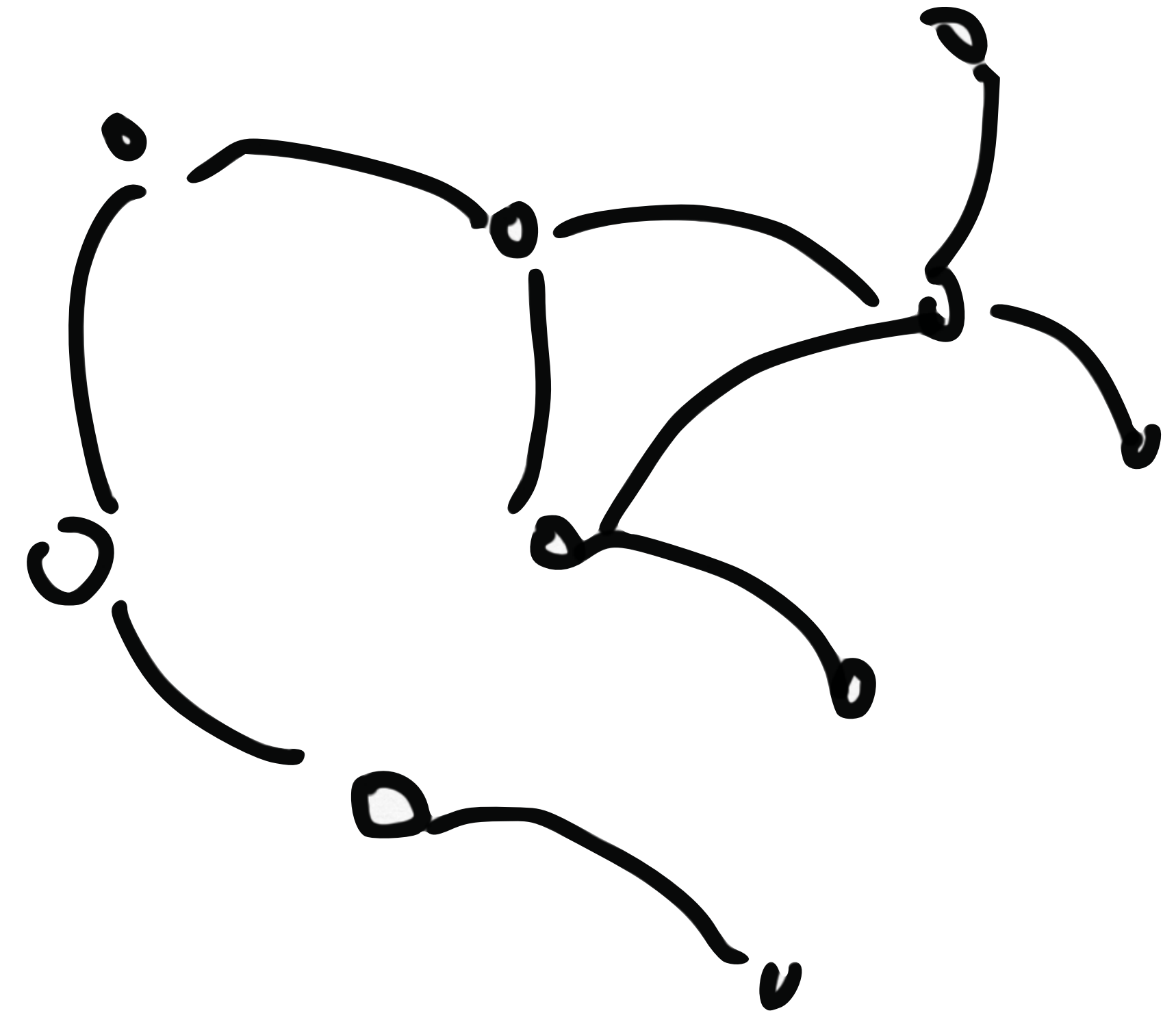
microservices are not the **goal**
they are the **means**

wishful mimicry

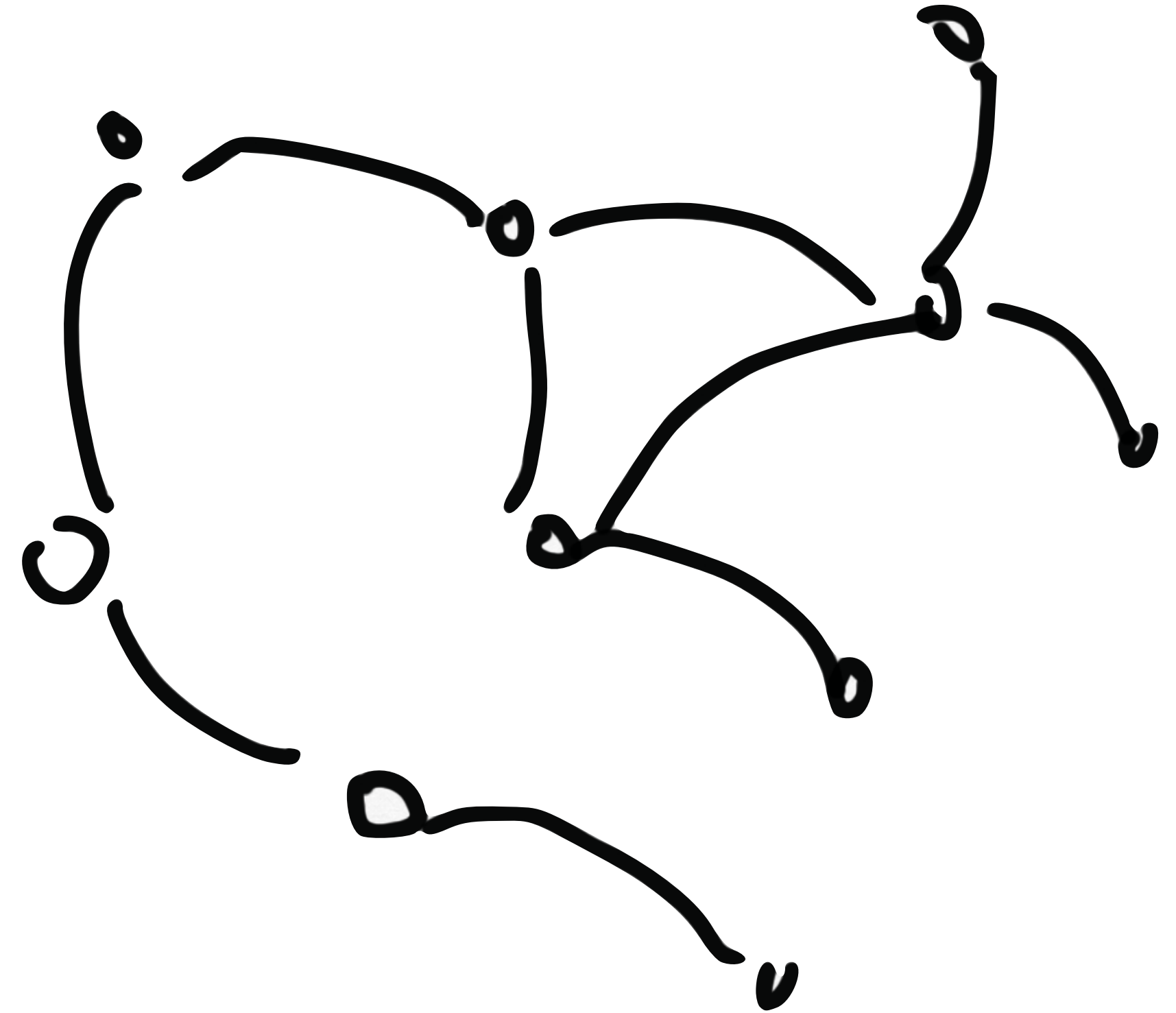
cloud != microservices

cloud native != microservices

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

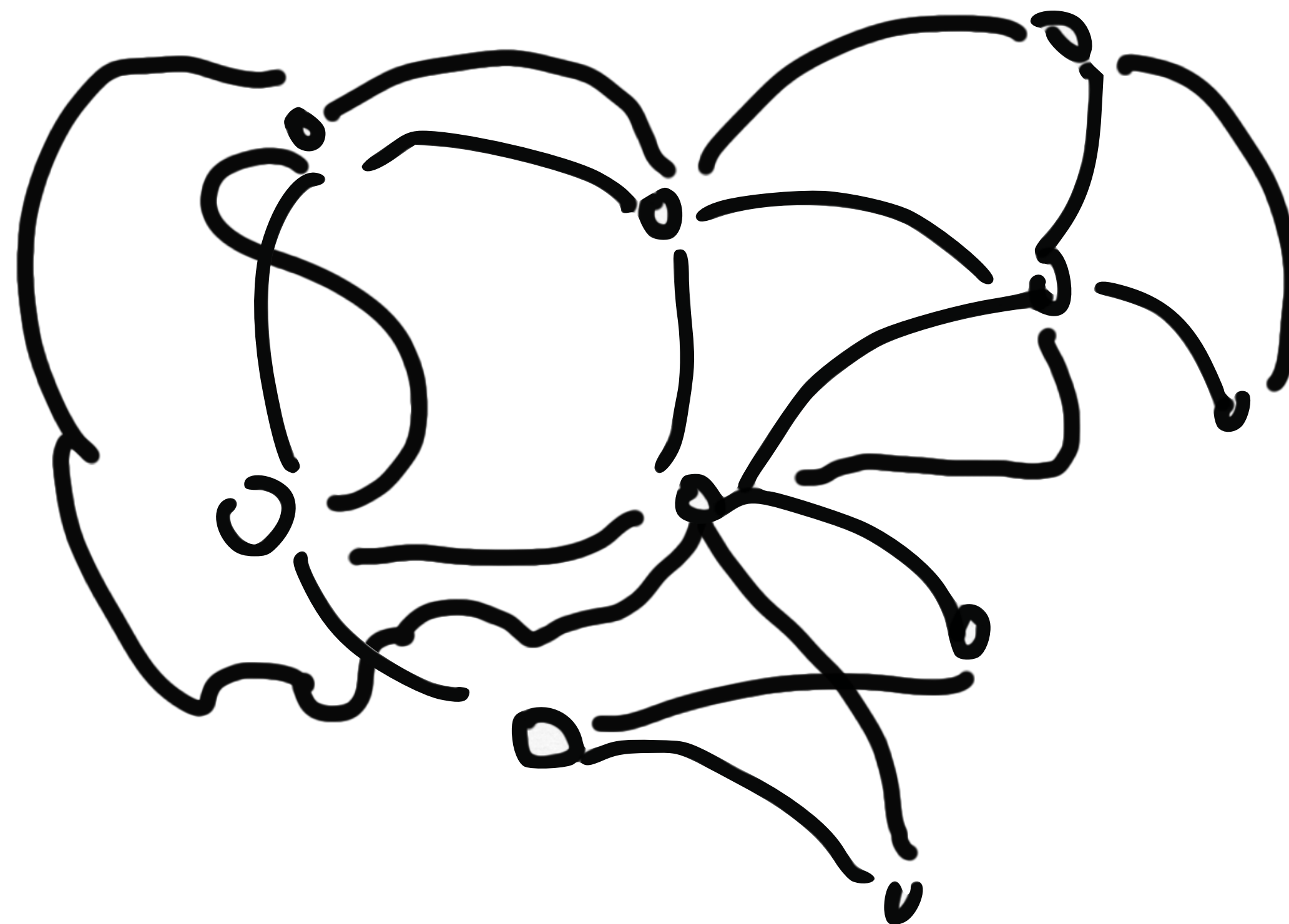


distributed monolith



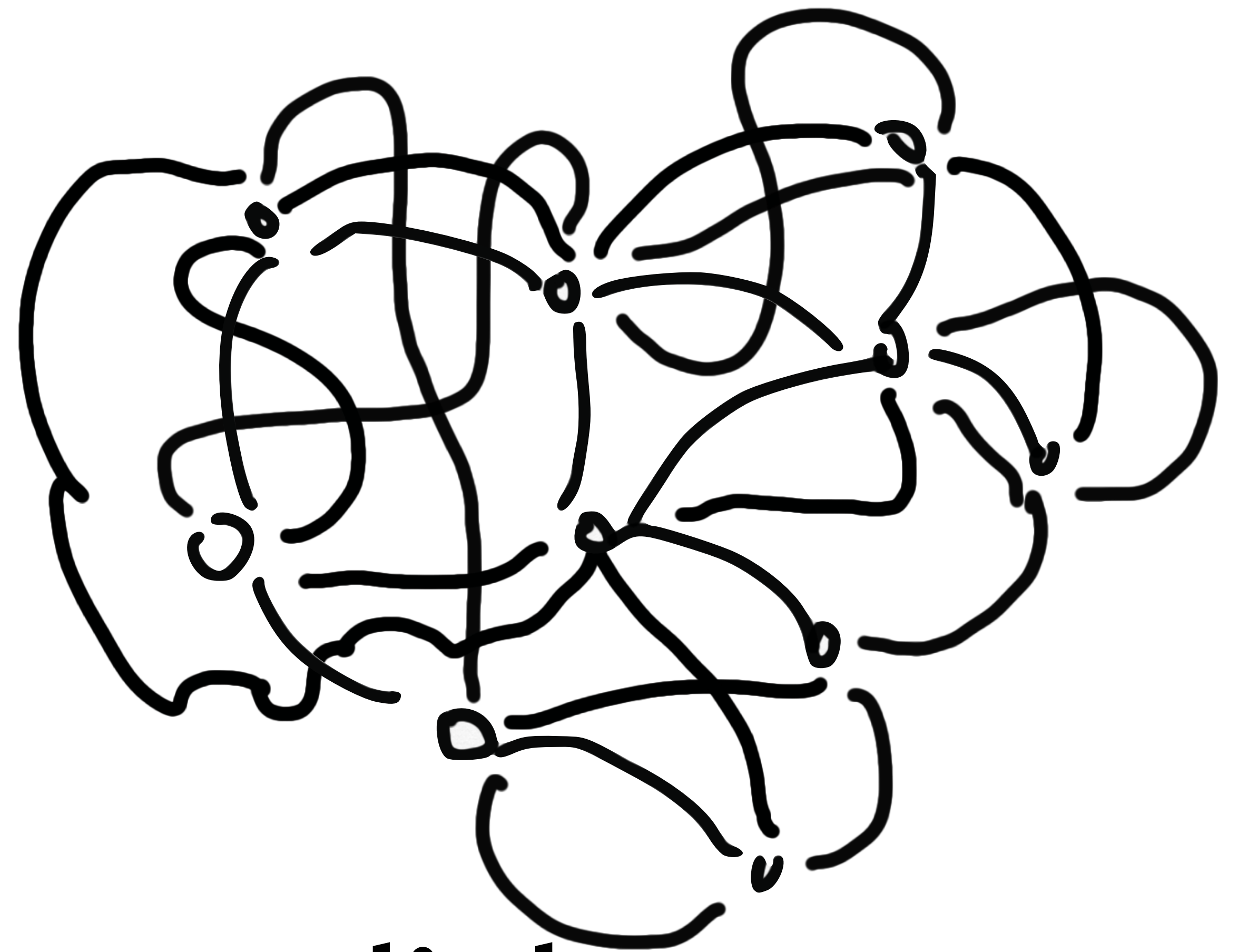
distributed monolith

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



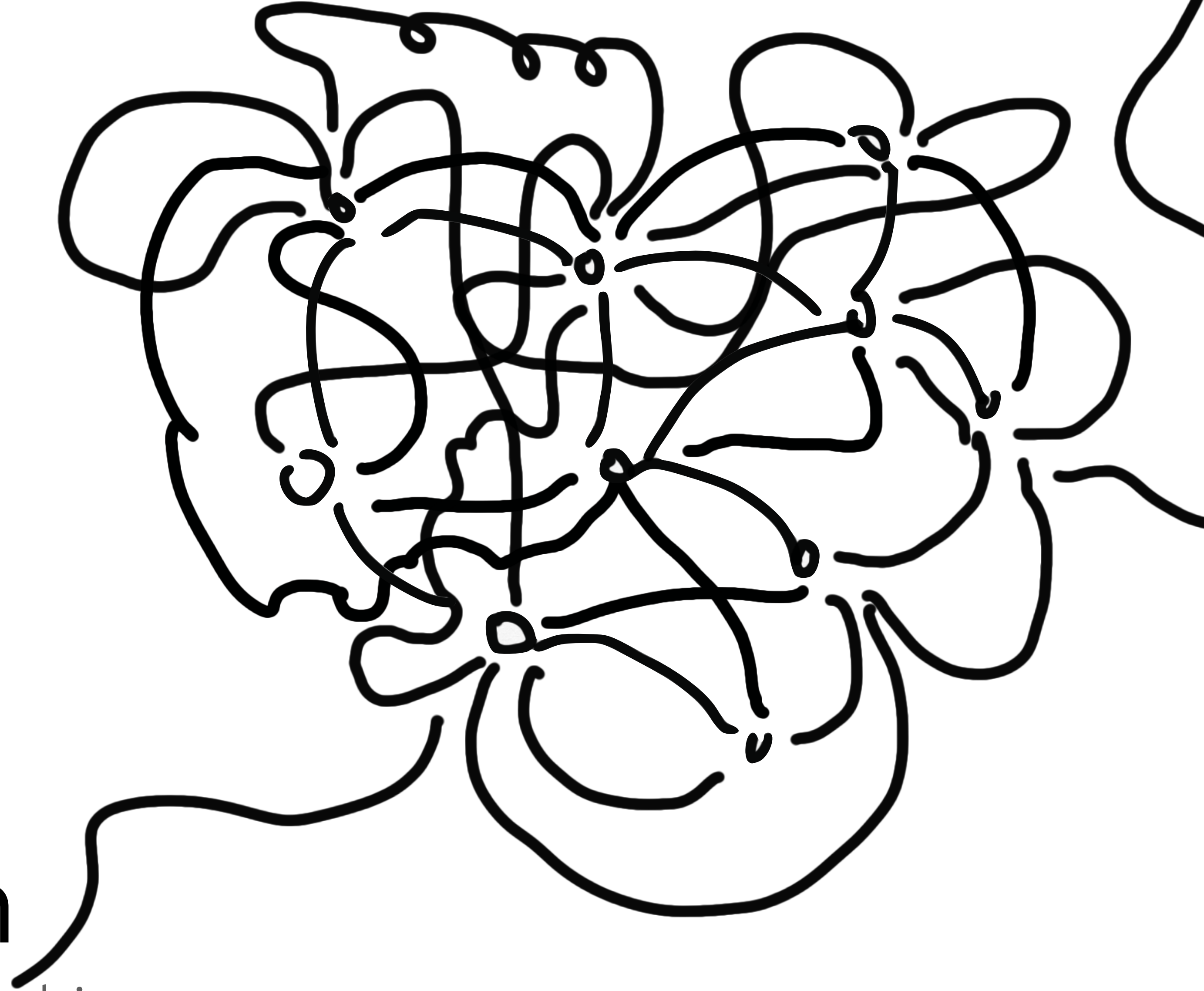
distributed monolith

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



distributed monolith

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

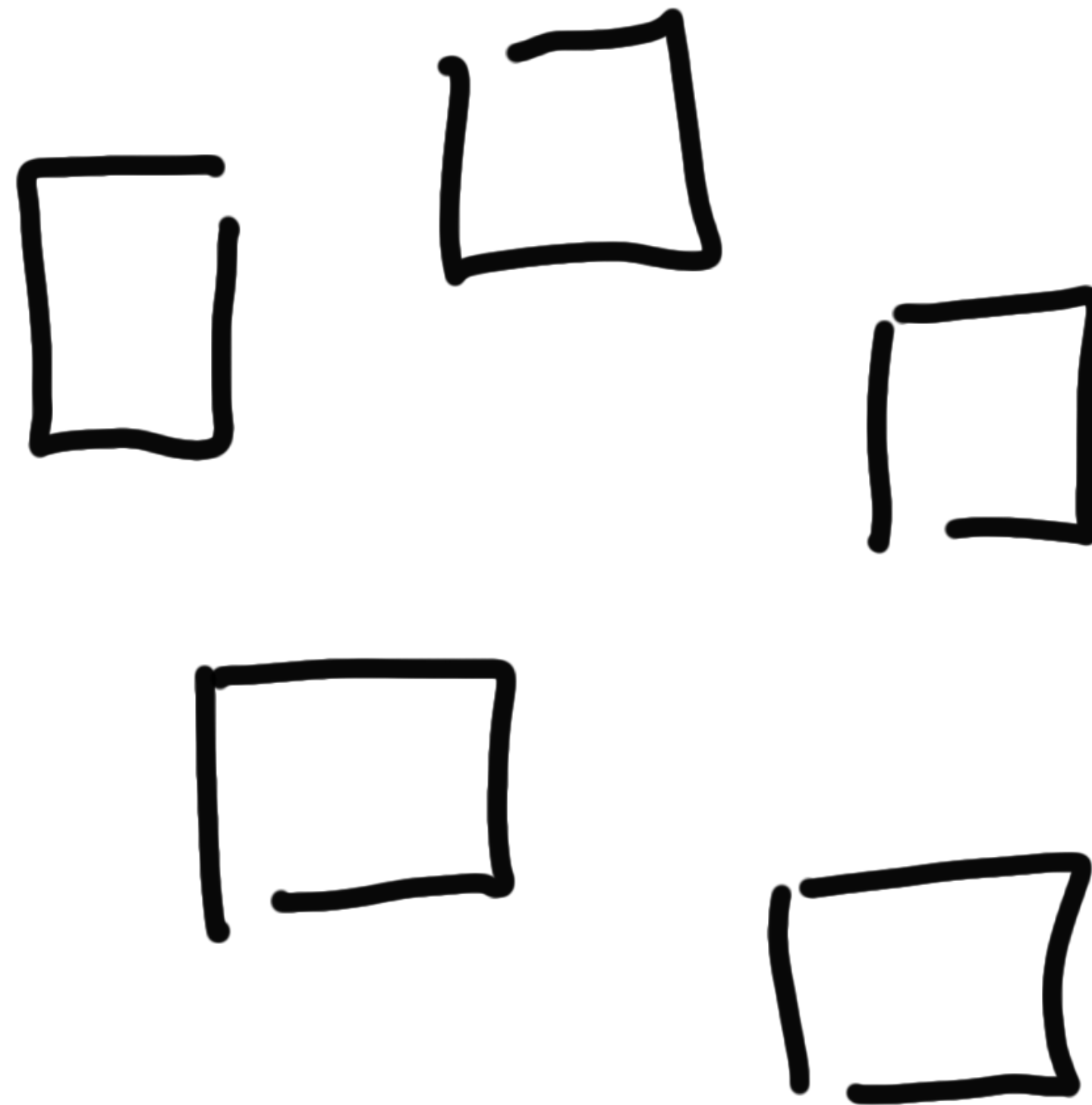


distributed monolith

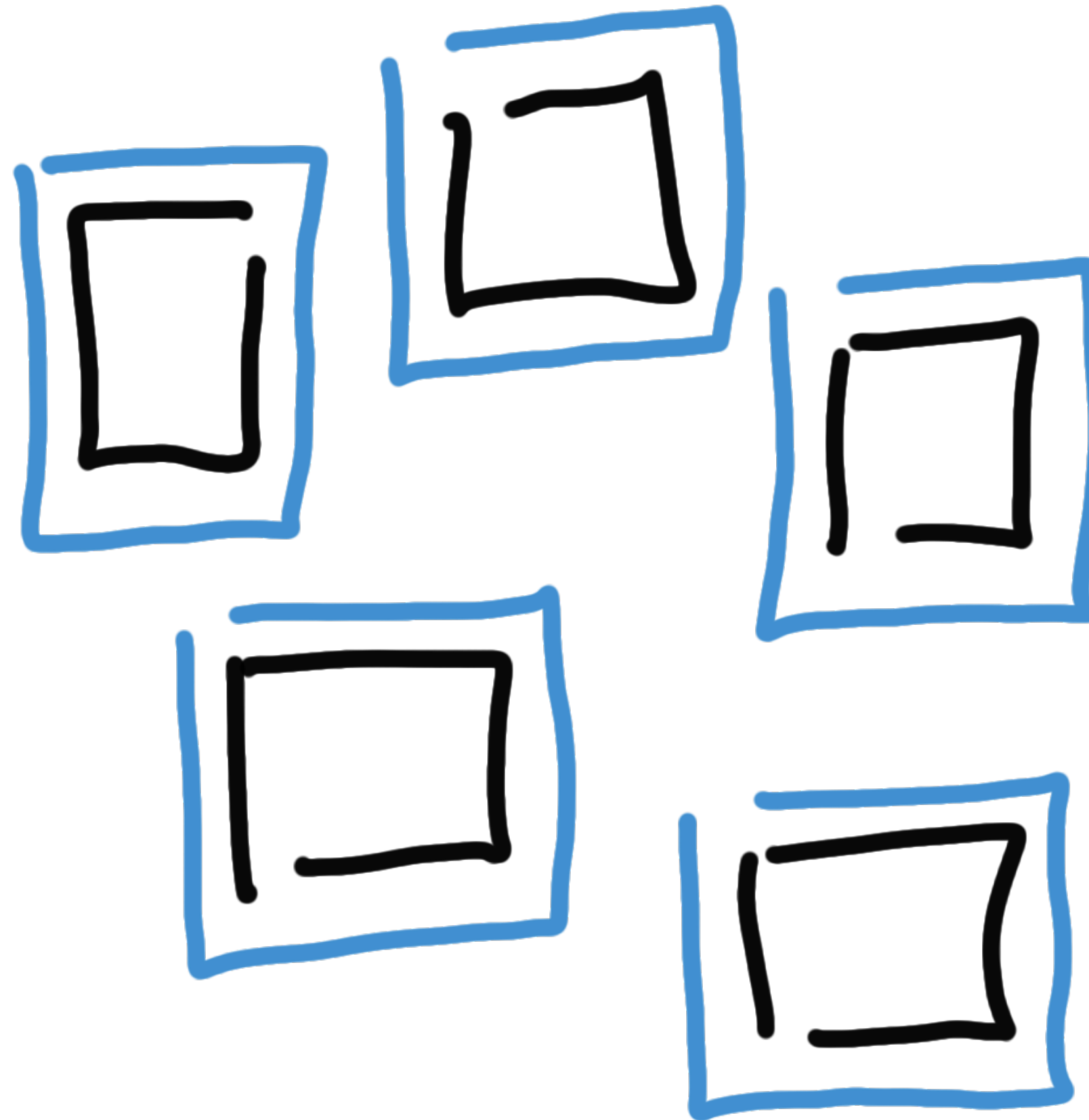
but without compile-time checking
... or guaranteed function execution

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

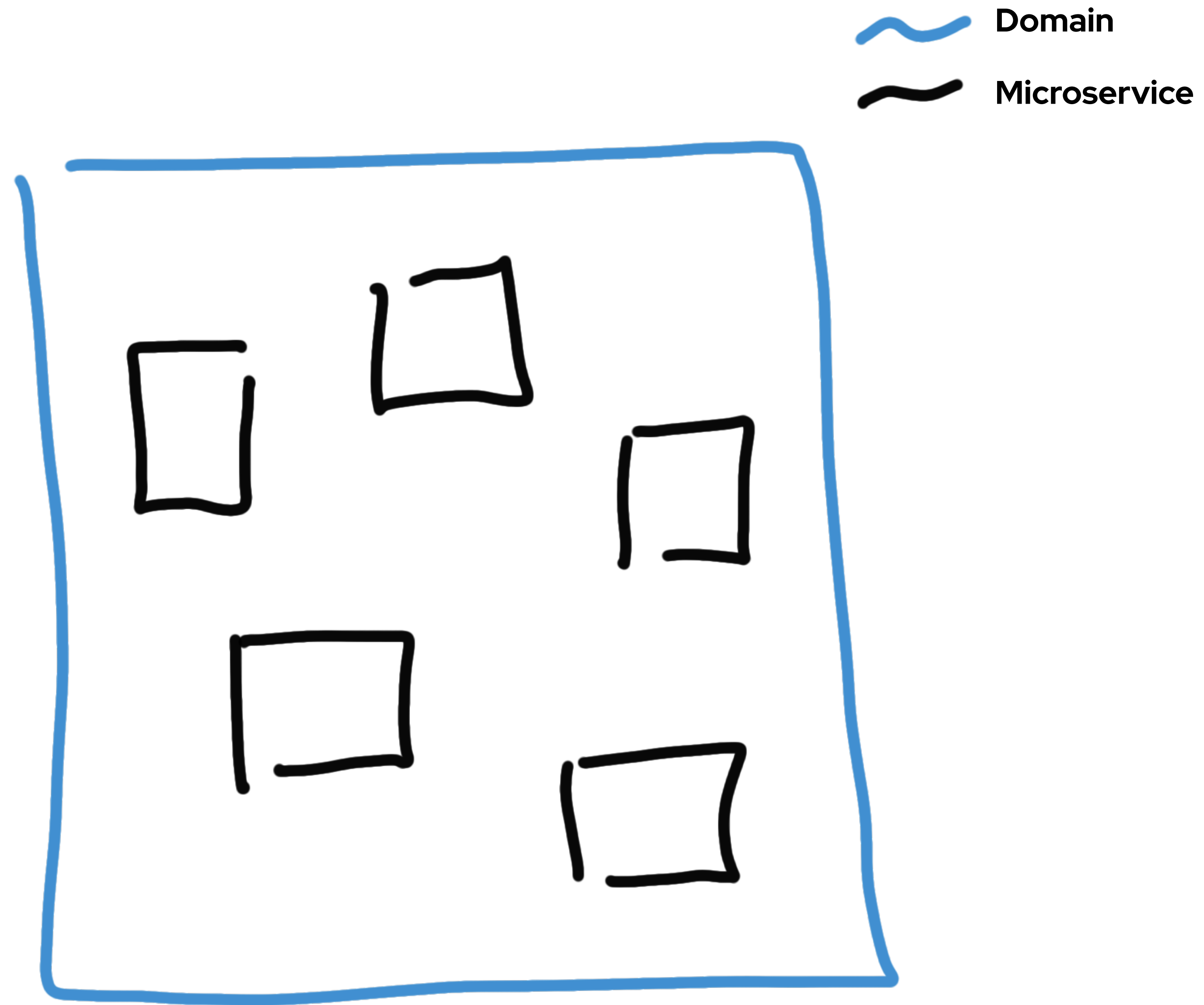
~ Microservice



Domain
Microservice



(this is bad)



distributed \neq decoupled

“ohhhh, we weren’t expecting your
service to do **that**...”

“uh, what do you mean you corrected the typo in your json?”

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

And testing.



the test
pyramid



the test pyramid

end-to-end tests



the test pyramid

end-to-end tests

unit tests



the test pyramid

end-to-end tests

contract tests

unit tests



the test pyramid

(you can TDD
at every level!)

end-to-end tests

contract tests

unit tests



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

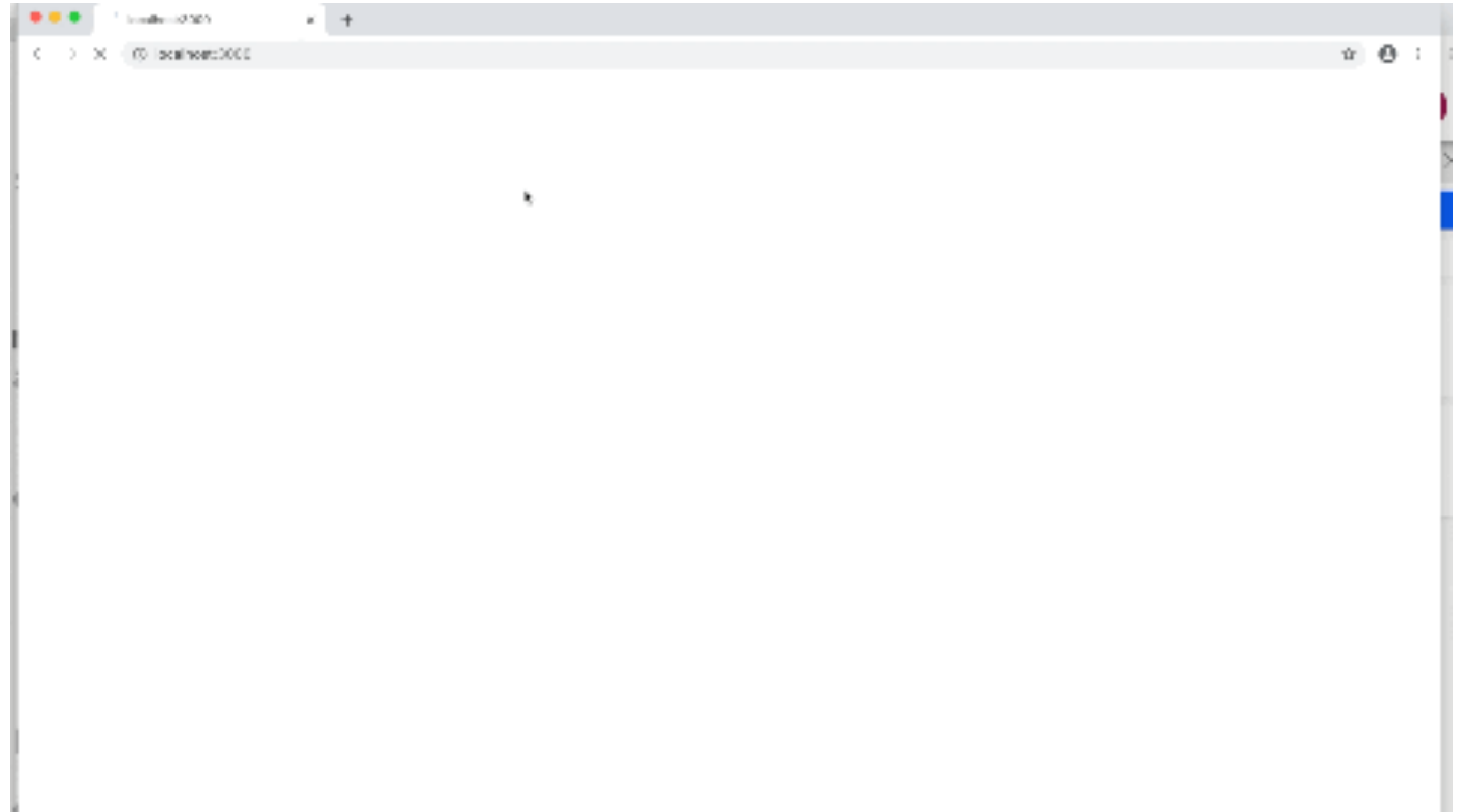


contract testing a
fire alarm

to the code!

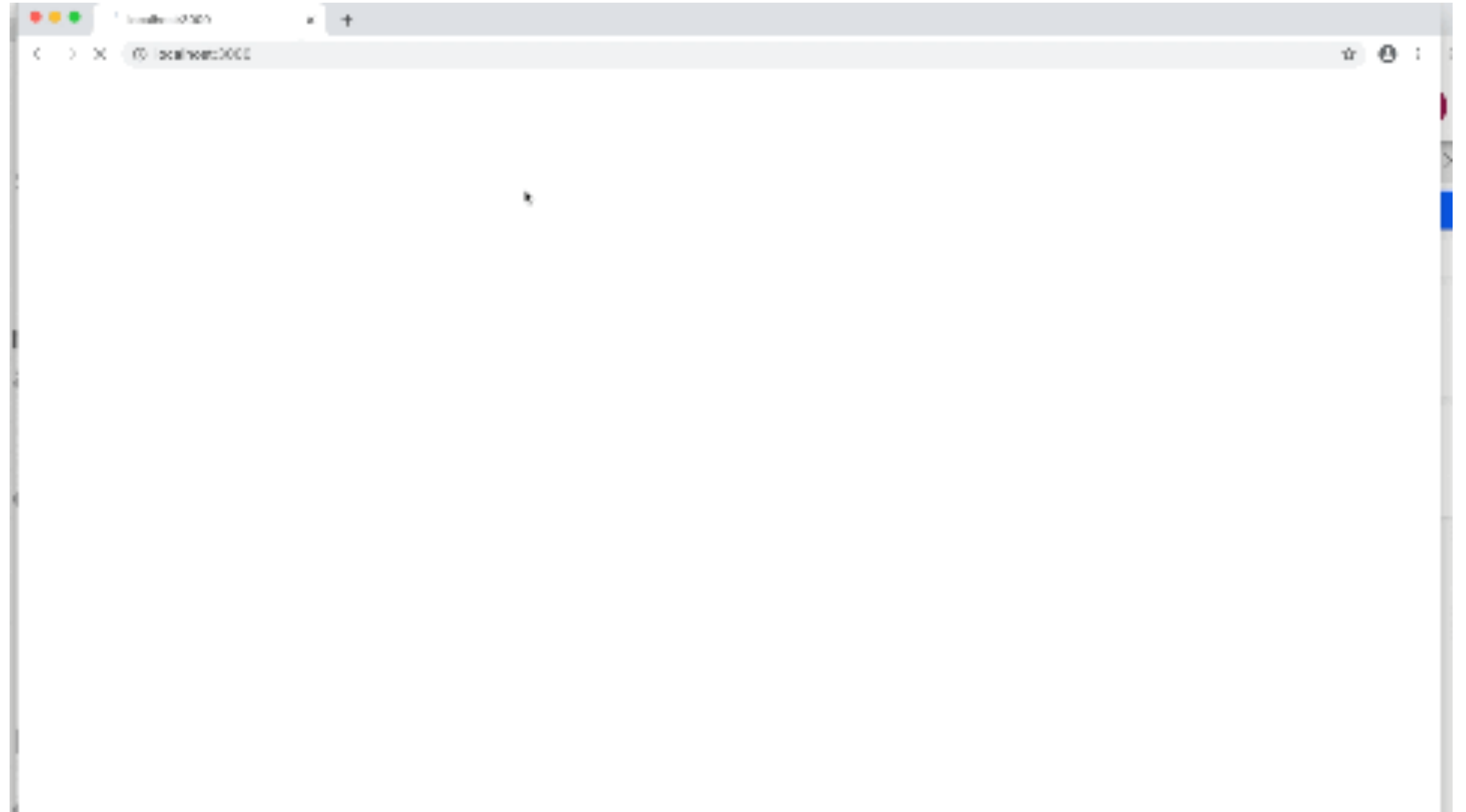
[illegible]

demo recap:



<https://github.com/holly-cummins/house-of-microservices-quarkus-contract-testing-sample>

demo recap:



<https://github.com/holly-cummins/house-of-microservices-quarkus-contract-testing-sample>

demo recap:

- consumer-driven contract testing can save your bacon
- pact is a mock for the consumer
- pact is a functional test for the producer
- shared json contracts aligns expectations across services

SO ...

don't assume cloud is a solved problem

don't assume microservices will solve all your problems

contract testing may solve some of your problems



slides



thank you

@holly_cummins



Red Hat

Don't forget to
rate this session
in the **GOTO Guide app**

slides →

